(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 1 355 485 A1

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
22.10.2003 Bulletin 2003/43

(51) Int Cl.[7]: **H04N 5/00**, H04L 12/28, H04L 29/00

(21) Application number: **02090147.6**

(22) Date of filing: **18.04.2002**

(84) Designated Contracting States:
AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE TR
Designated Extension States:
AL LT LV MK RO SI

(71) Applicant: **DEUTSCHE THOMSON-BRANDT GMBH**
78048 Villingen-Schwenningen (DE)

(72) Inventor: **Hütter, Ingo**
30982 Pattensen (DE)

(74) Representative:
Schäferjohann, Volker Willi, Dipl.-Phys. et al
Deutsche Thomson-Brandt GmbH
European Patent Operations
Karl-Wiechert-Allee 74
30625 Hannover (DE)

(54) Method for generating a user interface on a HAVi device for the control of a Non-HAVi device

(57) The invention relates to the bridging technology of home networks, namely a HAVi home network and an IP based network such as UPnP network. When combining both networks via a gateway (10) the service of controlling a UPnP device from a HAVi device shall be provided. A problem arises from the fact that not for every UPnP device a corresponding device control module exists in the HAVi network technology so that some of the UPnP devices cannot be controlled via a corresponding device control module (DCM).

The invention solves this problem by means of two basic software elements, namely a specialized function control module (FCM) to be implemented in the gateway (10) and a JAVA programme (HAVLET) that is run on the HAVi controller (31). The UPnP network is an IP based network. Therefore each UPnP device is represented by a so-called XML document. Such an XML document includes a number of XML descriptions, which are nothing else than function descriptions for the controllable elements. A function control module (FCM) according to the invention includes means for requesting the function descriptions of the UPnP device and for forwarding these function descriptions to the HAVi controller (31). The function control module (FCM) may include means for translating the retrieved function descriptions before forwarding to the HAVi controller (31). In the HAVi controller (31) the JAVA programme (HAVLET) is run and this programme takes the function descriptions received from the gateway (10) and generates a user interface with this information. The JAVA programme may be uploaded into the HAVi controller 31 during the configuration phase of the HAVi network.

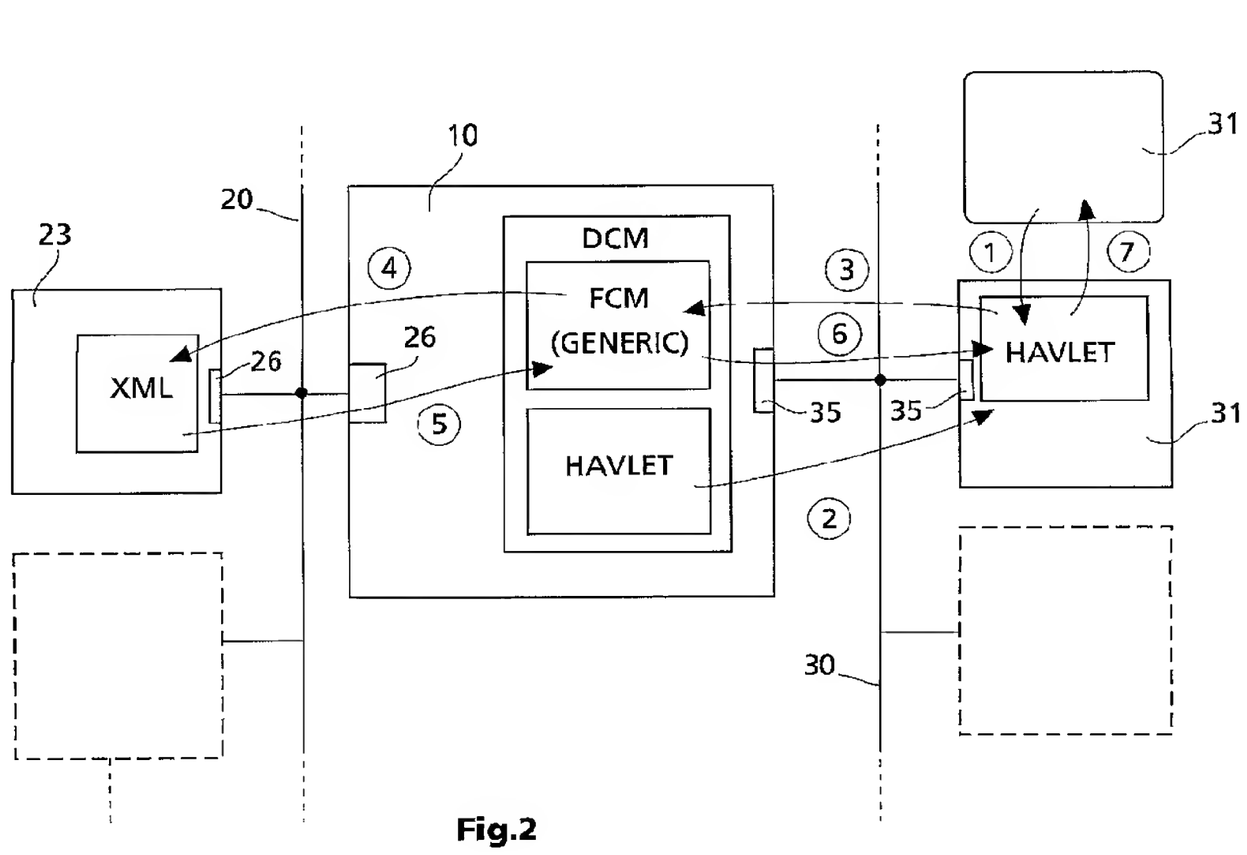

Fig.2

EP 1 355 485 A1

Printed by Jouve, 75001 PARIS (FR)

Description

[0001] The invention relates to a method for generating a user interface on a HAVi device for the control of a Non-HAVi device. This invention in particular applies to the field of domestic communication networks. The invention also concerns a gateway for use in the method for generating a user interface as well as two types of computer programme products.

Background

[0002] A few years ago the typical home audio/video equipment setup was characterized by a mix of CE devices of different types, e.g. a radio receiver, a CD player, a pair of speakers, a television set, a video cassette recorder, a tape deck, a DVD player, a satellite receiver and the like. For interaction of the devices point-to-point connection of analogue/digital input/outputs had to be made. For this purpose various kinds of different wires were available like Scart cables, Cinch cables, Coax cables, optical fibre, and so forth.

[0003] Meanwhile there are strong activities in the consumer electronics field to avoid this type of point-to-point connections. A number of standards for home networks already exist that can be used to connect all the different components to each other via a single type of network cable. In the consumer electronics field first of all the IEEE1394 bus standard should be mentioned here. The IEEE1394 bus system provides for a high data rate communication between the CE devices. The cable version supports data rates of 100, 200 and 400 Mbit/s. This is enough to transport asynchronous data for controlling a network station as well as isochronous audio and video streams in parallel. Isochronous and asynchronous data transfer modes are supported. However, the IEEE1394 standard specifies only the lower layers of the ISO/OSI reference model, namely the physical layer, the data link layer and the transaction layer. Therefore, the higher layers namely transport layer, session layer, presentation layer and application layer are left open for proprietary definition.

[0004] A consortium of consumer electronics companies worked on a standard for the audio/video electronics and the multimedia industry wherein the higher communication layers have been specified. This standard is referred to as the HAVi standard, where HAVi stands for Home Audio/Video interoperability. This standard primarily has defined an interoperability middleware that ensures that products of different vendors can interoperate, i.e. cooperate to perform application tasks. The application layer remains completely open to proprietary solutions.

[0005] Another consortium of companies, in particular computer companies including Microsoft, started another initiative for setting up a network control software stack based on Internet Protocols (IP). This network system is called UPnP (Universal Plug and Play) network. This system shall be open to all kinds of electronic components that could be integrated in a network in particular personal computers, but also electronic appliances in a household like refrigerators, microwave ovens, heating control, air conditioning control, security systems, washing machines and the like. The UPnP network system supports controlling of all these appliances via the Internet, therefore, even if somebody is on a journey, he can manage to monitor and control his home appliances.

[0006] Even though HAVi and UPnP have sometimes been seen as competitors, and in some ways they are, they serve somewhat different markets and have somewhat different goals. It is therefore foreseen a scenario that both networks exist in parallel in a household and that bridging is possible between the two for data exchange and interaction between UPnP network components and HAVi network components. This however calls for the creation of the bridging technology between HAVi and UPnP networks.

[0007] When talking of a bridge between UPnP and HAVi networks this technically means that a data packet is transferred to the other side on the data link layer. When transferring a data packet on a higher layer of the ISO/OSI reference model the bridging device is then called a gateway. As the data packets are transferred from HAVi to UPnP network or vice versa on a higher layer, the bridging device will be called gateway hereinafter. This is not meant limiting however.

[0008] With the gateway in between both networks, it shall be possible to control a HAVi device in the HAVi network from a UPnP device in the UPnP network. Also it shall be supported to control a UPnP device from a HAVi device.

[0009] For the controlling of the HAVi device from a UPnP device it is required to represent the HAVi devices as a UPnP device. Here, specific problems need to be solved that are not part of the present invention.

[0010] The invention deals with the problem of controlling a UPnP device from a HAVi device. To understand the invention it is advantageous to first explain the architecture of the HAVi system. According to the HAVi architecture a CE device in the network is controlled through abstract representations of the CE device. The architecture allows a module (e.g. device representation, controller, etc.) to send commands or control information to another module in the home network. A HAVi-compliant device contains data (above abstract representation referred to as device control module DCM) relating to its user interface and to its control capabilities. This data includes for example HAVi byte code (Java code) that can be uploaded and executed by other devices on the network. A HAVi-compliant device has, as a minimum, enough functionality to communicate with other devices in the HAVi network. During interaction, devices may exchange control data and application data in a peer-to-peer fashion. The HAVi specification distinguishes between controllers and controlled devices. A

controller is a device that acts as a host for a controlled device. A controller hosts the abstract representation for the controlled devices.

[0011] The HAVi specification defines HAVi compliant CE devices in the following categories: Full-AV devices (FAVs), Intermediate-AV devices (IAVs) and Base-AV devices (BAVs).

[0012] An FAV contains a complete set of software components of the HAVi software architecture. An FAV is characterized in that it has a run-time environment for HAVi byte code. This means it has a JAVA virtual machine. This enables an FAV device to download JAVA byte code form other devices, e.g. for providing enhanced capability for their control. An FAV may be formed by e.g. a HAVi compliant set top box, a HAVi compliant digital TV receiver or a home personal computer. For example an intelligent TV receiver can be the HAVi controller of other devices connected to the network. The receiver gets the byte code uploaded from another device in the network. An icon representing this device can be made to appear on the TV screen and user interaction with the icon may cause elements of the control programme to actuate the represented device in a pre-specified manner.

[0013] An IAV does not provide a run-time environment for HAVi byte code but may provide native support for control of specific devices on the home network. An IAV comprises embedded software elements that provide an interface for controlling general functions of the specific devices. These software elements need not be HAVi byte code and may be implemented as native applications on the IAV that use native interfaces to access other devices.

[0014] A BAV may provide uploadable HAVi byte code but does not host any of the software elements of the HAVi architecture. A BAV is controllable through an FAV by means of the former uploaded byte code. A BAV is controllable through an IAV via its DCM/FCM that has been uploaded by an FAV. Communication between an FAV or an IAV on the one hand and a BAV on the other hand requires that the HAVi byte code is instantiated by an FAV.

[0015] The HAVi specification includes a number of main software elements that are listed below. For a more detailed explanation of these elements it is referred to the HAVi specification. An existing version of the HAVi specification is V1.1, published May 15, 2001 and available from HAVi, INC., 2694 Bishop Drive, Suite 275 San Ramon, CA 94683, USA.

1. The 1394 communications media manager (CMM) acts as an interface between the other software elements and IEEE1394 bus.

2. An event manager (EM) informs the various software elements of events in the network such as the changes in the network configuration that occur when appliances (devices) are added or removed from the network.

3. A registry - maintains information about the appliances connected to the network and the functions they offer. Applications can obtain this information from the registry.

4. A messaging system (MS) - serves as an API (Application Programming Interface) that facilitates communication between the software elements of the various appliances on the network. The messaging system provides the HAVi software elements with communication facilities. It is independent of the network and the transport layers. The messaging system is in charge of allocating identifiers for the abstract representations at the FAV or IAV. These identifiers are first used by the abstract representations to register at the FAV or IAV. Then they are used by the abstract representations to identify each other within the home network. When a first abstract representation wants to send a message to another abstract representation it has to use the identifier of the latter while invoking the messaging API.

5. A device control module (DCM) - represents an appliance on the network. An application program can interact directly with a DCM. Within a DCM a number of functional control modules (FCM) may be contained. In a HAVi network, a functionality is represented by an FCM. Hierarchically speaking an FCM is always contained in a DCM, representing a device. A DCM may contain more than one FCM (e.g. a DCM representing a digital VCR contains a Tuner FCM and a VCR FCM) but there is only one DCM for each HAVi device.

6. A DCM manager - installs the DCMs. It automatically reacts to changes in the network by installing new DCMs for new BAV appliances.

7. A data driven interaction (DDI) controller - renders GUI (Graphical User Interface) on an appliance display on behalf of a HAVi software element. It supports a wide range of displays varying from graphical to text only.

8. A stream manager (SMGR) - creates connections and routes real time AV streams between two or more appliances on the network.

[0016] Basic HAVi interoperability addresses the general need to allow existing devices to communicate at a basic level of functionality. To achieve this, HAVi defines and uses a generic set of control messages that enable one device to communicate with another device and a set of event messages that it should reasonably expect from a device given its class (TV, VCR, DVD player, etc).

To support this approach a basic set of mechanism is required: Device discovery; Communication; and a HAVi message set. As to device discovery: Each device in the home network needs a well-defined method that allows it to advertise its capabilities to others. The HAVi approach is to utilize so-called SDD data: Self Describing Data. The SDD data is required on all HAVi devices in the network. SDD data contains information about the device, which can be accessed by other devices. The SDD data contains as a minimum enough information to allow instantiation of a so-called embedded device control module (embedded DCM). An embedded DCM is a piece of code preinstalled on a controlling IAV or FAV in a platform dependent code and using native interfaces to access the IAVs for FAVs resources. As mentioned above, a DCM for a device is a software element that provides an interface for control of general functions of the device. Instantiation of an embedded DCM results in registration of the devices capabilities with a registry. The registry provides a directory service and enables an object on the network to locate another object on the network. Registering allows applications to infer the basic set of command messages that can be sent a specific device on the network.

[0017] As to communication: Once an application has determined the capabilities of a device the application needs to be able to access those capabilities. This requires a general communication facility allowing applications to issue requests to devices. The service is provided by the HAVi messaging systems and DCMs. The application sends HAVi messages to DCMs, the DCM then engage in proprietary communication with the devices.

[0018] As to HAVI message sets: In order to support basic interoperability a well defined set of messages is required that must be supported by all devices of a particular known class (e.g. the class of TV receivers, the class of VCRs, the class of DVD players etc.). This ensures that a device can work with existing devices, as well as with future devices, irrespective of the manufacturer. These three basic requirements support a certain minimal level of interoperability. Since any device can query the capabilities of another device via the registry, any device can determine the message set supported by another device. Since applications have access to the messaging system, any device can interact with any other device.

[0019] Basic HAVi interoperability ensures that devices can interoperate at a basic level of functionality. However, a more extended mechanism is needed to also allow a device to communicate to other devices with any additional functionality that is not present in the embedded DCMs on an FAV. For example, embedded DCMs may not support all features of existing products and are unlikely to support those totally new ones of future product categories.

[0020] HAVi 'Level 2' interoperability provides this mechanism. To achieve this the HAVi architecture allows uploadable DCMs as an alternative to so called embedded device control modules. An uploadable DCM may be provided by any suitable source, but a likely technique is to place the uploadable DCM in the HAVI SDD data on the BAV device and upload from the BAV to the FAV device when the BAV is connected to the home network. Because the HAVi architecture is vendor-neutral it is necessary that the uploaded DCM will work on a variety of FAV devices all with potentially different hardware architectures. To achieve this, uploaded DCMs are implemented in HAVi (JAVA) byte code. The Java byte code run-time environment on FAV devices supports the instantiation and execution of uploaded DCMs. Once created and running within an FAV device the DCM communicates with the BAV devices in the same manner as described above.

[0021] Under the new scenario, where a HAVi network is connected with a UPnP network via a gateway, and a UPnP device shall be controlled by a HAVi FAV device, the additional problem occurs that none of the UPnP devices provides a HAVi DCM that could be uploaded to the HAVI FAV device. Therefore, neither basic nor level 2 interoperability is available for the controlling of UPnP devices from a HAVi network station.

Invention

[0022] It is an object of the invention to solve the problem of controlling a Non-HAVi compliant device in a Non-HAVi network from a HAVi compliant FAV device in a HAVi network via a gateway. For some of the UPnP devices there could exist in the gateway device a corresponding representation in the form of a DCM with an embedded device specific FCM. This DCM/FCM could be used for generating a user interface on the HAVi FAV device for controlling the UPnP device using basic interoperability. The user could, therefore, generate control commands for the UPnP device that need to be interpreted in the gateway and transformed into a corresponding UPnP command that would be understood in the UPnP device to be controlled.

[0023] A problem is however that there are certainly UPnP devices existing for which no corresponding representation in the form of a FCM, exist in the HAVi system. For such a case there is the possibility implemented in the HAVi system to generate a so-called generic FCM. In case of an unknown UPnP device, the gateway can only provide a DCM having embedded a generic FCM for the control of the UPnP device. With this generic FCM the HAVi FAV device cannot generate a user interface because none of the functions of the UPnP device are known in the generic FCM. This is the crux of the problem underlying the invention.

[0024] The invention solves the problem with the means of the independent claims 1, 8, 12 and 15. The invention utilizes the possibility in the HAVi system to download from a DCM a so-called HAVLET that is executable JAVA byte code to generate a user interface on

the HAVi controller. This HAVLET software piece interacts with the DCM for the Non-HAVi devices stored and executed in the gateway. The Non-Havi DCM contains a specialized Non-HAVi FCM that includes the software routines for requesting the function descriptions of a Non-HAVi device and for forwarding them to the HAVI FAV device. The HAVLET running on the HAVi FAV device takes the function descriptions of the Non-HAVi device and generates a corresponding user interface with these function descriptions.

[0025] Advantageous, modifications and improvements of the invention are listed in the dependent claims. Very advantageous is, if the FCM running on the gateway comprises means for translating the function descriptions read from the Non-HAVi device (23) into a data form supported by the HAVi system before forwarding to the HAVi controller. This improvement simplifies the HAVLET software running on the HAVi controller very much. The means for translating the function description of the Non-HAVi device need not be included in the HAVLET, thus making it unnecessary to upload a corresponding software code into the HAVi FAV device, thereby reducing memory requirements in the HAVI FAV device. Likewise the processor of the HAVi FAV device is relieved.

[0026] The invention can best be utilized if a HAVi network needs to be combined with an IP-based network, e.g. the UPnP network. In case of a UPnP network a UPnP device is represented by means of so-called XML descriptions for each function of a UPnP device. The XML descriptions will be requested by the specialised function control module which is of the type generic FCM (according to the HAVi specification) running on the gateway, translated and then be forwarded to the HAVLET executed by the HAVi controller. For each translated function description the HAVLET will generate a graphical representation preferably in the form of a button, slider, query button or input field together with a symbol or expression that explains the meaning.

[0027] A gateway according to the invention is claimed in independent claim 8.

[0028] Independent claim 12 claims a computer programme product, namely a function control module FCM for the gateway according to the invention.

[0029] Independent claim 15 claims a computer programme product, in particular HAVLET, running on the HAVi controller according to the invention.

Drawings

[0030] Exemplary embodiments of the invention are illustrated in the drawings and are explained in more detail in the following description.

[0031] In the figures:

Fig. 1 shows an example of a HAVi network and a UPnP net-work connected to each other via a gateway;

Fig. 2 shows the basic software elements interacting to each other of the UPnP device, the gateway and the HAVi controller;

Fig. 3 shows an example of a user interface for the control of a UPnP security camera displayed on the HAVi controller;

Fig. 4 shows a programme listing for a function control module for the use in the gateway according to the invention;

Fig. 5 shows a programme listing for a HAVLET to be executed by a HAVi controller according to the invention; and

Fig. 6 shows a programme listing for a service description routine that will be called when executing the function control modul.

Exemplary embodiments of the invention

[0032] Fig. 1 depicts the principle structure of two networks being connected to each other via a gateway 10. On the left side of the figure it is shown a UPnP network. As an example reference number 21 denotes a washing machine, reference number 22 a refrigerator, reference number 23 a security camera, reference 24 a heating control unit, and reference 25 a personal computer having an ISDN/DSL Internet connection. All these UPnP devices are connected to an Ethernet data bus 20 for data exchange. The Ethernet bus lines are also connected with gateway 10. On the right side of figure 1 is shown a HAVi network. The reference number 31 marks a TV set, reference number 32 denotes a VCR, reference number 33 denotes a DVD player and reference number 34 stands for a set top box such as a digital satellite receiver. The HAVi network stations are connected to an IEEE1394 bus 30 for data exchange. The gateway 10 is also connected to the 1394 bus 30. The gateway 10 comprises an IP protocol stack 11, on one side, a HAVi protocol stack 12 on the other side as well as software for carrying out the translation or mapping of control messages and events from one network to the other.

[0033] The HAVi as well as the UPnP specifications are known in the art. Therefore, there is no need to explain all the details in these specifications for the purpose of disclosing the present invention. It is therefore expressively referred to the HAVi specification as well as the UPnP specification for this purpose. The UPnP specification can be obtained from the UPnP Forum managed by Microsoft Incorporation.

[0034] As mentioned before the UPnP network system is based on the existing Internet protocols. A graphical user interface (GUI) for controlling UPnP devices from an UPnP controller, e.g. the personal computer 25 may consist of a plurality of icons displayed on the computer monitor. When a user selects an icon, the HTML pages are retrieved from of the device in question. The HTML pages are displayed for the user. This allows the user to control the given device. In the UPnP specification it is defined that each UPnP device comprises a list

of services, which are provided by the device. Each of these services is described in an XML document, where XML stands for Extension Mark-up Language, i.e. Internet technology. Each XML document contains a detailed description of all control possibilities within the service. These XML documents will be utilized for controlling a UPnP device from a HAVi controller.

[0035] The control process of the UPnP device from a HAVi FAV device is illustrated in Fig. 2. Identical reference numbers denote the same components as shown in Fig. 1 and need not be explained again. In Fig. 2 the Ethernet interface circuit 26 with which UPnP devices as well as the gateway are equipped are separately shown. Similarly the 1394 bus interface 35 for the HAVi network components and the gateway 10 are likewise shown. In addition the basic software elements of the security camera 23, the gateway 10 and the TV set 31 are shown in Fig. 2. A security camera 23 contains an XML document in which the control possibilities for the security camera are listed. The important software element of the gateway is a device control module DCM containing a specified function control module FCM as well as an executable JAVA programme HAVLET. The JAVA programme HAVLET is provided for an upload to a HAVi FAV device during the configuration phase of the HAVi network. Therefore, the very important software element of the HAVi controller 31 concerns this HAVLET.

[0036] For controlling the security camera 23 the gateway interacts with the security camera 23 and the TV set 31 as follows.

[0037] After finishing of the configuration phase in both networks, all the network components within the HAVi network as well as in the UPnP network can be controlled from the TV set 31. A user interface for controlling these devices is built in the form of a list of icons for each controllable device. It is the communications media manager CMM, the event manager EM, the registry and the messaging system MS of the HAVi protocol stack, that are utilized for collecting the information of all controllable elements be it in the HAVi network or in the UPnP networks. Of course, the gateway 10 includes corresponding software elements and interfaces so that the mapping of the UPnP devices in the HAVi registry is possible. This process however is pre-supposed to be prior art and will not be explained in further detail here.

[0038] A user may now wish to control the security camera from the TV set 31 in the HAVi network. For this purpose he selects the corresponding icon on the TV screen. This event will start the download of the HAVLET into the internal memory of the TV set 31. Right after the download, the execution of the HAVLET is started. The HAVLET is an executable JAVA programme. As JAVA is a platform independent programming language, it will run on each HAVi FAV device that has a run-time environment for JAVA byte code.

[0039] Third, the executed HAVLET sends a request for retrieving information about the security camera 23 to the gateway 10. This request will be accepted by the running UPnP function control module FCM that itself retrieves the XML document/s stored on the webserver of the security camera 23. Each XML document contains descriptions of the control possibilities for the security camera 23. The FCM translates the XML descriptions into a struct (a set of variables) and forwards them to the HAVLET running in the TV set 31. The HAVLET then takes these function descriptions and generates for each controllable element a graphical representation such as button, slider, query button, input field or the like to generate the graphical user interface for the control of the security camera on the TV screen. The flow of information is illustrated in Fig. 2 with arrows and the numbering expresses the order of interaction.

[0040] The graphical user interface for the security camera 23 is shown in Fig. 3. For each controllable element of the security camera a graphical representation is generated, e.g. for the brightness setting a slider is shown on the TV screen. With a mouse pointer the brightness can directly be controlled by means of the left/right buttons positioned at the left and right side of the slider. Also the slider itself could be dragged to the wanted position as known from a great variety of computer menus. At the left side of the slider for the brightness adjustment the expression Set-Brightness is shown in writing. This expression is directly taken over from the XML description for this controllable element. The FCM running in the gateway does not necessarily know the meaning of this expression. This is evident if one considers that a new type of product can be integrated in the UPnP network for which today nobody knows what the controllable elements are. In such a case the user has to make the right interpretation of this controllable element by himself. Below the brightness slider a GetBrightness button is positioned. This is an example of a query button. By depressing this query button the current brightness set value will be read out and displayed beside the button. Below the GetBrightness button are positioned simple buttons Increase-Brightness and DecreaseBrightness. These buttons have the same effect as the right and left buttons of the SetBrightness slider. An example of an input field is the field SetDefaultRotation. Here a number is requested with this field and shall be entered into the input mask. The input parameter determines the rotation of the security camera for getting another view. Instead of displaying the extracted expression for the respective control element derived from the XML description, a self-explanatory symbol could be displayed in the user interface. This however calls for the need to have pre-defined user interface components installed in the HAVLET for the different services of the various possible UPnP devices. Even if a new type of appliance will be integrated in the UPnP network, the pre-defined user interface component could be used if this new device provides a service for which the user interface components are already provided in the HAVLET. For unknown services however, this solution would not work. Another solution is that

there will be a mix of both different solutions for one service. For all the parameters in the XML description to which already a symbol had been assigned, a corresponding symbol could be displayed in the user interface. For the unknown parameters however, the corresponding expressions need to be shown.

[0041] The XML document of a UPnP device can be regarded as a standard implementation of the UPnP specification. For the realization of the present invention no extraordinary programming has to be made here. The basic software elements for realization of the present invention are the UPnP FCM running on the gateway and the HAVLET uploaded into the HAVi FAV device. Both software elements include extraordinary routines for the realization of the invention.

[0042] Fig. 4 shows the programme listing of the function control module for the UPnP network. It is somehow a 'generic' FCM that will always be used for the control of any UPnP device. The programming language is JAVA. This is a well-known programming language widely used so that the particular syntax need not be explained here. The important routines for implementing the invention are labelled. With label Ⓐ the routine for extracting the services from the XML document is marked. This routine therefore extracts which kind of service the requested UPnP device offers. For example the UPnP security camera offers the service of providing a stream of video pictures in a particular format like JPEG or Image at a certain compression level and resolution.

[0043] The GET_SERVICE_DESCRIPTION routine marked with label Ⓑ requests the information how many services the UPnP device offers.

With the GET_SERVICE_INFORMATION_LIST routine marked with label © the information about every control possibility of the selected service can be retrieved. The routine PERFORM_CONTROL_COMMAND is provided for sending a control command to a UPnP device, see label Ⓓ. The routine PERFORM_DEVICE_VARIABLE_QUERY is provided for retrieving the current variable value from a UPnP device, see label Ⓔ. These routines will be executed upon a request from the HAVLET that runs on the HAVi device.

[0044] For the routine call corresponding instructions are hold in the second part of the programme listing headed methods for answering the incoming request. Expressively it is referred to the routine call DO_GET_SERVICE_DESCRIPTION marked with label Ⓕ, and DO_GET_SERVICE_INFORMATION_LIST marked with label Ⓖ. A further important routine for the implementation of the invention is the routine sendControlCommand for sending a control command to the UPnP device. This routine is marked with label Ⓗ. Within this routine a response message is also evaluated and forwarded to the HAVLET in the HAVi device. Also important for the implementation of the invention is the routine queryDeviceVariable marked with label Ⓘ. This routine is started if the HAVLET has sent a corresponding request. For example if the user has pressed a query button, this routine will be called. Again in this routine a response message will be sent back to the HAVi device. With the routine receiveHttpNotifyData marked with label Ⓙ UPnP events will be handled.

[0045] The JAVA source code of a HAVLET implementing the invention is shown in Fig. 5. The main task of the HAVLET is to build the user interface for controlling a UPnP device. The complete routine for building the UI is labelled Ⓚ. The HAVLET contains corresponding routines for getting the service descriptions for a UPnP device marked with label Ⓛ, for getting the service information list marked with label Ⓜ, for performing a control command marked with label Ⓝ and for performing a device variable query marked with label Ⓞ.

[0046] The function control module is integrated in a device control module according to the HAVi specification. Therefore what needs to be done is to programme a device control module having embedded the function control module of Fig. 4. This programme is regarded to be a standard implementation of the HAVi specification that needs not to be explained in detail. That is why the listing of the DCM is not being shown.

[0047] The part of the programme that performs the translation of the XML descriptions into a data form supported by the HAVi system, is included in a routine called service description routine, shown in Fig. 6. The XML descriptions are basically in text format. These XML descriptions are evaluated. For example the programme part marked with label Ⓟ evaluates whether the XML descriptions contain some actions. These actions correspond to the controllable elements of the UPnP device. They are translated into the HAVi-typical form of a variable set called 'struct'. This is performed by parsing the XML description and storing all information of interest in its local instance variables.

Claims

1. Method for generating a user interface on a HAVi device for the control of a Non-HAVi device, where HAVi stands for Home Audio/Video interoperability, the HAVi device (31) being a station of a HAVi network and the Non-HAVi device (23) being a station of a Non-HAVi network, both networks being connected to each other by a gateway (10), **characterized in that** the gateway (10) runs a function control module (FCM) for the Non-HAVi devices that requests the descriptions of the functions of the Non-HAVi device (23) to be controlled and forwards them to the HAVi device (31) that generates the corresponding user interface components for the functions of the Non-HAVi device (23) by means of a JAVA programme (HAVLET) that runs on the HAVi device (31).

2. Method according to claim 1, wherein the function control module (FCM) translates the function descriptions read from the Non-HAVi device (23) into a data form supported by the HAVi system before forwarding to the HAVi device (31).

3. Method according to claim 1, wherein the JAVA programme (HAVLET) translates the function descriptions read from the Non-HAVi device (23) into a data form supported by the HAVi system.

4. Method according to one of claims 1 to 3, wherein the gateway (10) uploads the JAVA programme (HAVLET) to the HAVi device (31) during configuration.

5. Method according to one of the previous claims, wherein the HAVi device (31) on which the user interface for controlling the Non-HAVi device (23) is generated is a HAVi device of the FAV type, wherein FAV means Full Audio/Video HAVi device.

6. Method according to one of the previous claims, wherein the Non-HAVi network is an IP based network, in particular a UPnP network, where UPnP stands for the Universal Plug and Play network system.

7. Method according to claim 6, wherein the function descriptions of the Non-HAVi device (23) are XML descriptions, where XML stands for Extension Mark-up Language.

8. Gateway for use in a method according to one of the previous claims, comprising an interface (35) for a HAVi network and comprising an interface (26) for a Non-HAVi network, **characterized in that** the gateway (10) comprises a function control module (FCM) that includes means for requesting the function descriptions of the Non-HAVi device (23) and means for forwarding the function descriptions to the network station of the HAVi network on which a user interface for controlling a Non-HAVi (23) device shall be generated.

9. Gateway according to claim 8, comprising a JAVA programme (HAVLET) that includes means for generating a user interface with the function descriptions of a Non-HAVi device (23), this JAVA programme (HAVLET) being provided for an upload into a HAVi device (31).

10. Gateway according to claim 8 or 9, wherein the function control module (FCM) comprises means for translating the function descriptions read from the Non-HAVi device (23) into a data form supported by the HAVi system before forwarding to the HAVi network.

11. Gateway according to claim 9, wherein the JAVA programme (HAVLET) comprises means for translating the function descriptions read from the Non-HAVi device (23) into a data form supported by the HAVi system

12. Computer programme product, in particular function control module (FCM) directly loadable into the internal memory of a gateway (10) according to one of the claims 8 to 11, comprising means for requesting function descriptions of a Non-HAVi device (23) and means for forwarding the function descriptions to a network station of a HAVi network on which a user interface for controlling the Non-HAVi device (23) shall be generated when said product is executed by a processor of the gateway (10).

13. Computer programme product according to claim 12, further comprising means for translating the function descriptions read from the Non-HAVi device (23) into a data form supported by the HAVi system before forwarding to the HAVi network.

14. Computer programme product according to claim 12 or 13, wherein the function descriptions of the Non-HAVI device (23) are XML descriptions, where XML stands for Extension Mark-up Language.

15. Computer programme product, in particular JAVA programme (HAVLET), directly loadable into the internal memory of a HAVi device (31) of a HAVi network, comprising means for generating a user interface for the control of a Non-HAVi device (23) with the function descriptions retrieved from the Non-HAVi device (23), when said product is executed by a processor of said HAVi device (31).

16. Computer programme product according to claim 15, wherein the retrieved function descriptions are translated XML descriptions of the Non-HAVi device (23), where XML stands for Extension Mark-up Language and the XML descriptions being translated into a data form supported by the HAVi system.

17. Computer programme product according to claim 15, further comprising means for translating the function descriptions retrieved from the Non-HAVi device (23) into a data form that is supported by the HAVi system and the function descriptions are XML descriptions, where XML stand for Extension Mark-up Language.

18. Computer programme product according to one of claims 15 to 17, wherein the means for generating a user interface comprise means for assigning to a translated function description the appropriate graphical representation together with a symbol or expression that explains the meaning of the func-

tion description.

19. Computer programme product according to claim 18, wherein the graphical representation is in the form of a button, slider, query button or input field.

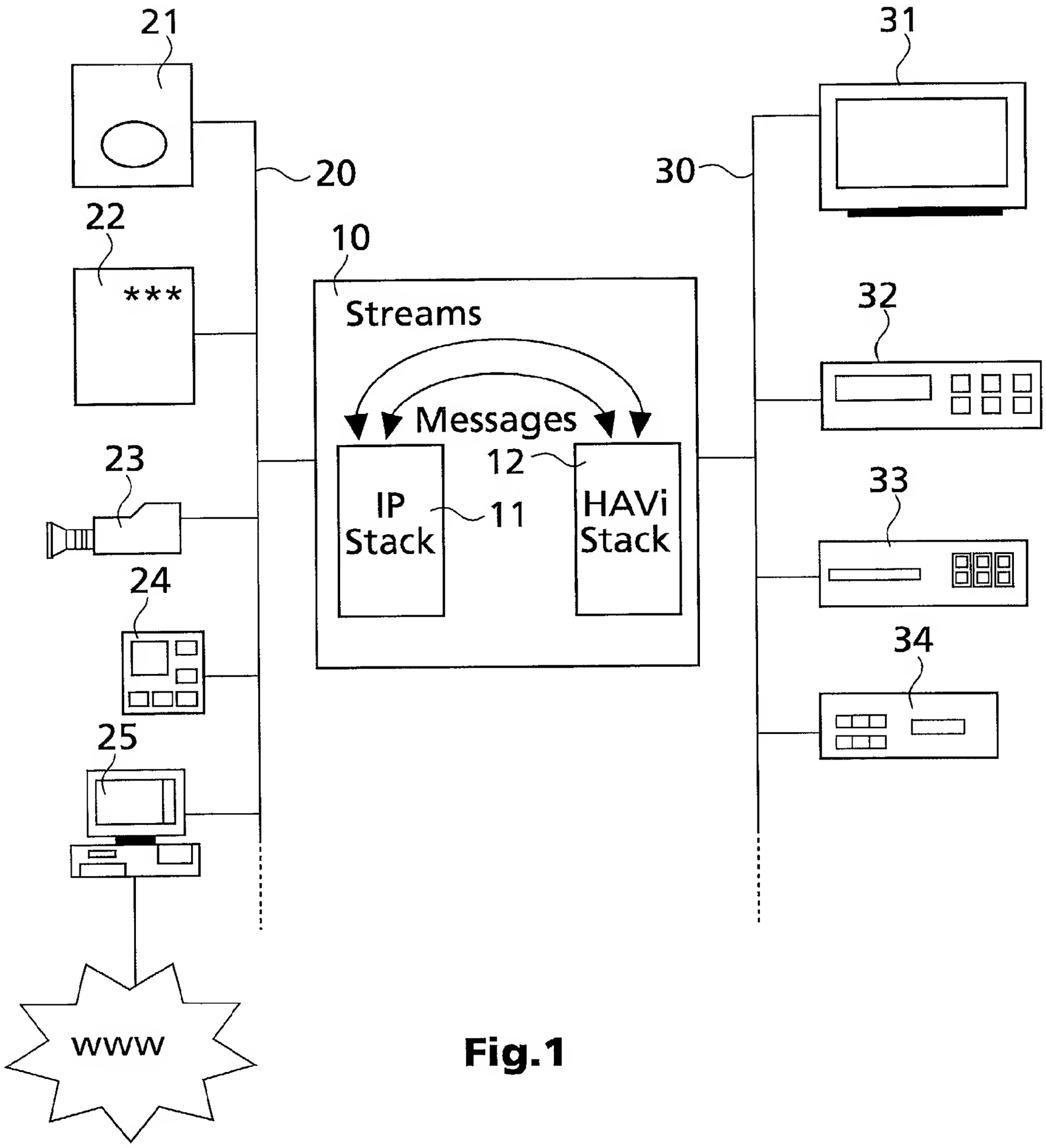
21
20
22
***
10
Streams
Messages
23
12
IP
Stack
11
HAVi
Stack
24
25
www
31
30
32
33
34

Fig.1

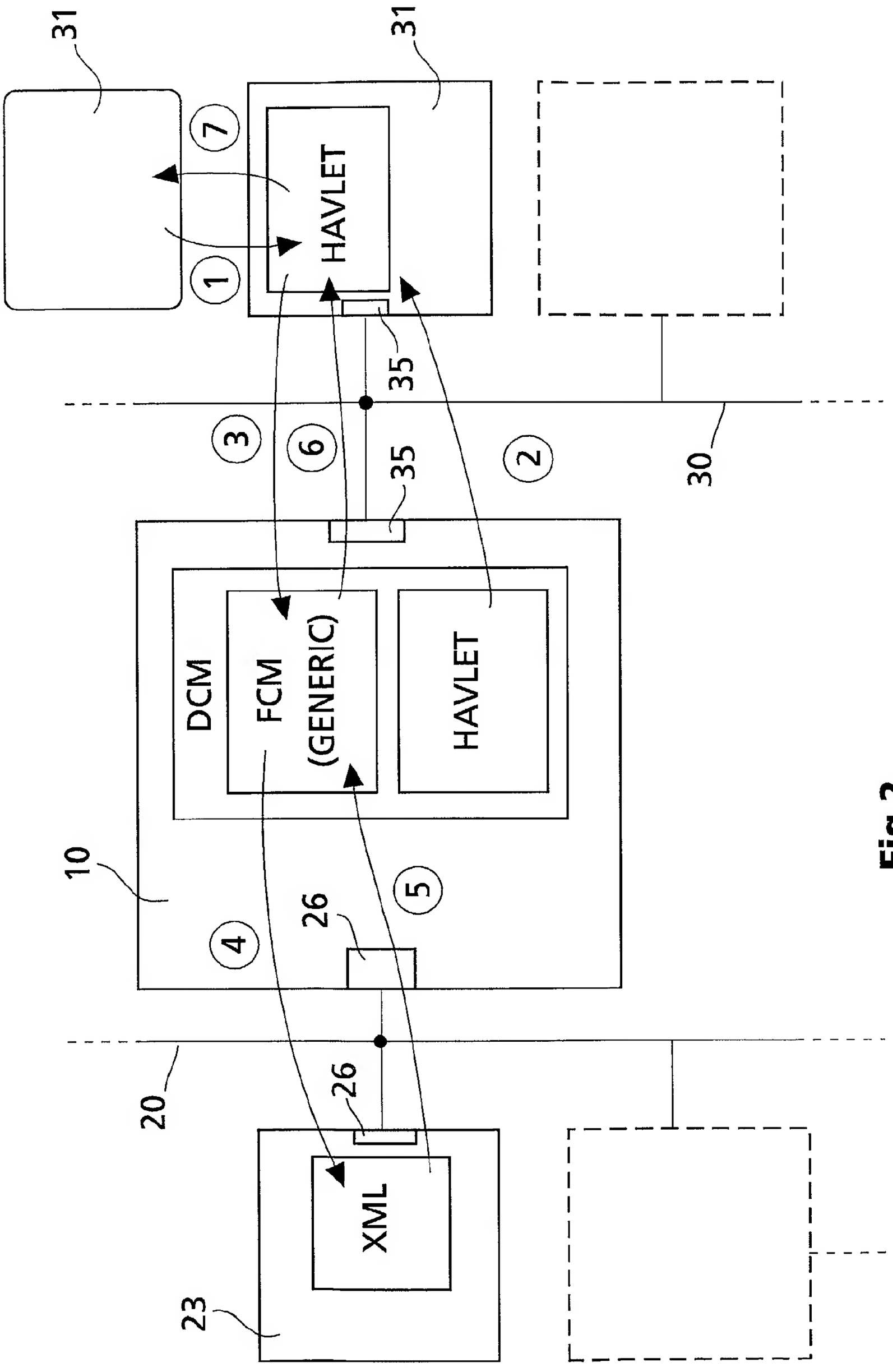
31
31
7
1
HAVLET
35
3
6
35
2
30
DCM
FCM
(GENERIC)
HAVLET
10
5
4
26
26
20
XML
23

Fig.2

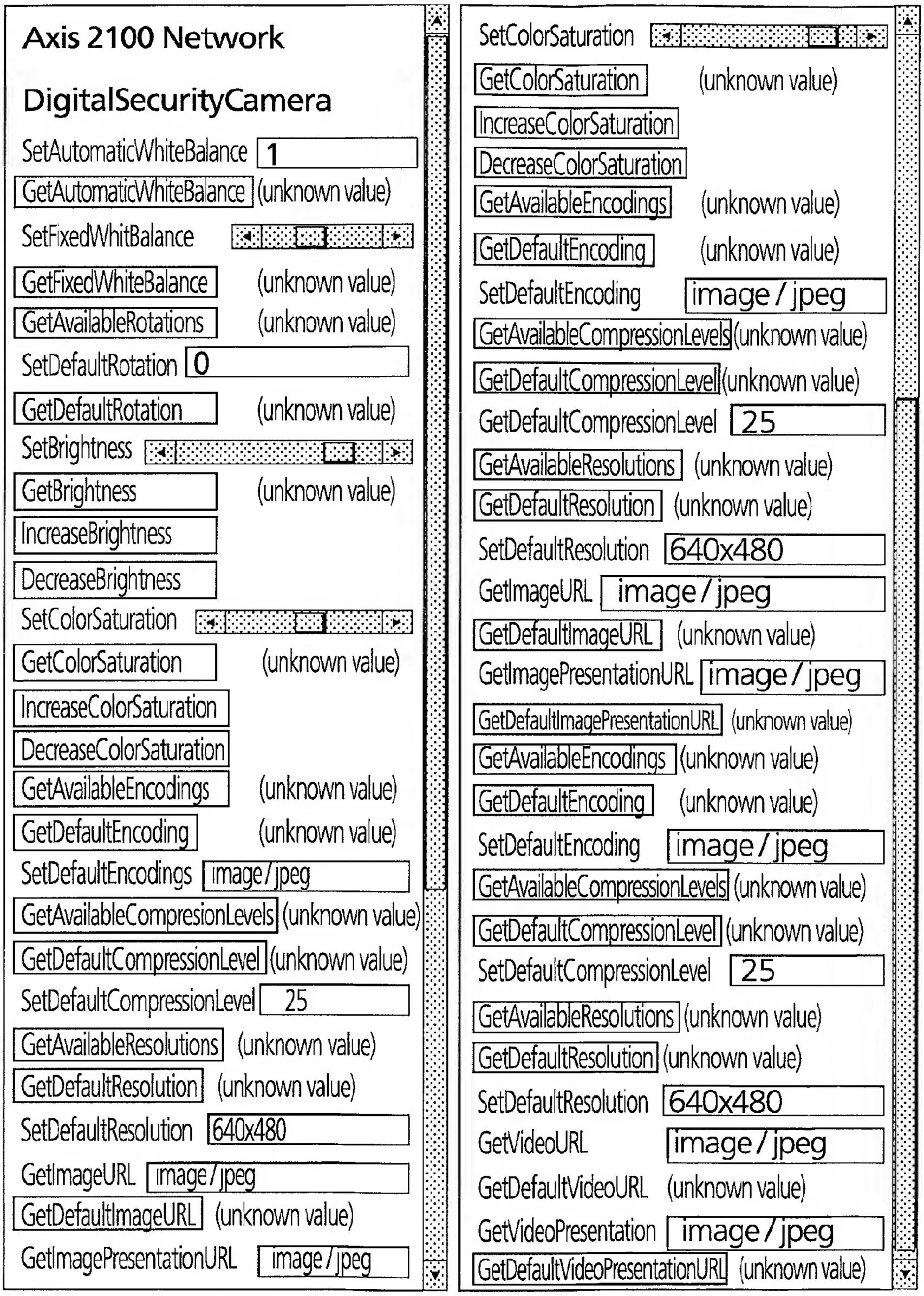
Axis 2100 Network
DigitalSecurityCamera
SetAutomaticWhiteBalance 1
GetAutomaticWhiteBalance (unknown value)
SetFixedWhitBalance
GetFixedWhiteBalance (unknown value)
GetAvailableRotations (unknown value)
SetDefaultRotation 0
GetDefaultRotation (unknown value)
SetBrightness
GetBrightness (unknown value)
IncreaseBrightness
DecreaseBrightness
SetColorSaturation
GetColorSaturation (unknown value)
IncreaseColorSaturation
DecreaseColorSaturation
GetAvailableEncodings (unknown value)
GetDefaultEncoding (unknown value)
SetDefaultEncodings image/jpeg
GetAvailableCompresionLevels (unknown value)
GetDefaultCompressionLevel (unknown value)
SetDefaultCompressionLevel 25
GetAvailableResolutions (unknown value)
GetDefaultResolution (unknown value)
SetDefaultResolution 640x480
GetImageURL image/jpeg
GetDefaultImageURL (unknown value)
GetImagePresentationURL image/jpeg
SetColorSaturation
GetColorSaturation (unknown value)
IncreaseColorSaturation
DecreaseColorSaturation
GetAvailableEncodings (unknown value)
GetDefaultEncoding (unknown value)
SetDefaultEncoding image/jpeg
GetAvailableCompressionLevels (unknown value)
GetDefaultCompressionLevel (unknown value)
SetDefaultCompressionLevel 25
GetAvailableResolutions (unknown value)
GetDefaultResolution (unknown value)
SetDefaultResolution 640x480
GetImageURL image/jpeg
GetDefaultImageURL (unknown value)
GetImagePresentationURL image/jpeg
GetDefaultImagePresentationURL (unknown value)
GetAvailableEncodings (unknown value)
GetDefaultEncoding (unknown value)
SetDefaultEncoding image/jpeg
GetAvailableCompressionLevels (unknown value)
GetDefaultCompressionLevel (unknown value)
SetDefaultCompressionLevel 25
GetAvailableResolutions (unknown value)
GetDefaultResolution (unknown value)
SetDefaultResolution 640x480
GetVideoURL image/jpeg
GetDefaultVideoURL (unknown value)
GetVideoPresentation image/jpeg
GetDefaultVideoPresentationURL (unknown value)

Fig.3

```
//-----------------------------------------------------------------------
// Copyright (c) 2000-2001, This software is the property of Thomson
// multimedia, and shall not be reproduced, copied, or distributed
// without written permission.
//-----------------------------------------------------------------------

//-----------------------------------------------------------------------
// TITLE:         FcmUpnp - Implementation of the FcmUpnp class.
// AUTHOR(s):     Ingo Huetter
// DATE CREATED: 21.09.2001
// FILENAME:      FcmUpnp.java
// PROJECT:       DHN (HAVi implementation)
// COPYRIGHT:     (C) 2000-2001 THOMSON multimedia,
//  THIS SOFTWARE IS THE PROPERTY OF THOMSON multimedia, AND SHALL NOT
//  BE REPRODUCED, COPIED, OR DISTRIBUTED WITHOUT WRITTEN PERMISSION.
//
// VERSION: 1.1
//
// MODIFICATION HISTORY:
//    DATE MODIFIED:
//    AUTHOR(s):
//    MODIFICATION MADE:
//    PROBLEM CAUSING MODIFICATION:
//
// DESCRIPTION:
//  Base class of all FCMs for UPnP devices. If an UPnP device can be mapped to
//  an HAVi device, a special FCM for this kind of devices can be created (eg.
//  FcmUpnpDisplay) derived from FcmUpnp. If mapping is not possible, a
//  FcmUpnpNonhavi is derived from FcmUpnp.
//
//-----------------------------------------------------------------------

package com.thmulti.havi.upnp;

import org.havi.constants.*;
import org.havi.types.*;
import org.havi.system.*;
import org.jdom.*;
import org.jdom.input.*;
import org.jdom.output.*;
import java.io.*;
import java.net.*;
import java.util.*;


public class FcmUpnp
extends java.lang.Object
implements ReceiveHttpNotifyInterface
{
    // --- Variables ---
    FcmListener         fcm_listener;
    SoftwareElement     fcm_se;
    FcmServerHelper     fcm_server;

    HUID                my_huid;
    TargetId            my_target_id;
    VendorId            my_vendor_id;
    short               my_fcm_index;
    SEID                my_dcm_seid;
    String              my_user_preferred_name;
    String              my_device_manufacturer;
    String              my_device_type;
    short               my_interface_id;
    int                 my_device_class;
```

```
    SAXBuilder          builder;
    Vector              service_list;
    UpnpWebServer       my_upnp_web_server;

    // --- Constructor ---
    public FcmUpnp(UpnpWebServer upnp_web_server, String device_type)
    {
        my_upnp_web_server = upnp_web_server;
        my_device_type     = device_type;
        builder            = new SAXBuilder();
        service_list       = new Vector();
    }

    // --- install ---
    public int install(Element xml_device, Namespace ns, String base_url, SEID
dcm_seid, short fcm_index, String user_preferred_name, String
device_manufacturer, SEIDHolder fcm_seidh)
    {
        // save parameters
        my_fcm_index            = fcm_index;
        my_dcm_seid             = dcm_seid;
        my_user_preferred_name  = user_preferred_name;
        my_device_manufacturer  = device_manufacturer;
        my_device_class         = ConstDeviceClass.LAV;

        // Create the listener and the software element
        try {
            fcm_listener = new FcmListener();
            fcm_se       = new SoftwareElement(fcm_listener);
            fcm_seidh.setValue(fcm_se.getSeid());
        } catch (Exception e) {
            fcm_se.log(fcm_se.MSG_ERROR, fcm_se.MSG_SENDER_FCM,
"FcmUpnp::install: Creating fcm_listener or fcm_se failed!");
            return 1;
        }

        // Build the HUID
        try {
            my_target_id = new TargetId(ConstTargetType.FCM_NON61883, new
GUID(fcm_se.getSeid().getValue()), 0, my_fcm_index); // CHANGE LATER
            my_vendor_id = new VendorId(fcm_se.getSeid().getValue());
            my_huid = new HUID(my_target_id, (short)0, my_vendor_id, false, 0);
        } catch (Exception e) {
            fcm_se.log(fcm_se.MSG_ERROR, fcm_se.MSG_SENDER_FCM,
"FcmUpnp::install: Creating HUID failed!");
        }

        // Create the FcmServerHelper
        try {
            fcm_server = new FcmServerHelper(fcm_se, ConstTransferMode.SIMPLE);
        } catch (HaviGeneralException e) {
            fcm_se.log(fcm_se.MSG_ERROR, fcm_se.MSG_SENDER_FCM,
"FcmUpnp::install: Creating fcm_server failed!");
            return 1;
        }

        // Extract the services from the XML device Description
        Element xml_service_list = xml_device.getChild("serviceList",ns);
        if (xml_service_list != null) {
            java.util.List xml_services =
xml_service_list.getChildren("service",ns);
            if (xml_services.size() > 0) {
                for (int i=0;i<xml_services.size();i++) {
                    Element xml_service = (Element)xml_services.get(i);
```

```
                    ServiceDescription sd = new ServiceDescription();
                    if (sd.init(xml_service, ns, base_url) == false) {
                        fcm_se.log(fcm_se.MSG_WARNING, fcm_se.MSG_SENDER_FCM,
"FcmUpnp::install: Fetching service information failed!");
                        break;
                    }
                    service_list.addElement(sd);
                }
            }
        }

        // Subscribe for events

my_upnp_web_server.registerClassHttpNotify(getUpnpEventNotificationPath(),
this);
        for (int i=0;i<service_list.size();i++) {
            ServiceDescription sd =
(ServiceDescription)service_list.elementAt(i);
sd.sendEventSubscription("http://"+my_upnp_web_server.getHostAddress()+":"+my_up
np_web_server.getPort()+getUpnpEventNotificationPath(), 30000);
        }

        fcm_se.log(fcm_se.MSG_INIT, fcm_se.MSG_SENDER_FCM, "FcmUpnp was
installed");

        return 0;
    }

    // --- uninstall ---
    public void uninstall()
    {
        // Unsubscribe for events
        for (int i=0;i<service_list.size();i++) {
            ServiceDescription sd =
(ServiceDescription)service_list.elementAt(i);
            sd.sendEventUnsubscription();
        }

my_upnp_web_server.unregisterClassHttpNotify(getUpnpEventNotificationPath());

        // Clear the ServiceList
        service_list.clear();

        // Unregister from the MessagingSystem
        try {
            fcm_se.close();
        } catch (Exception e) {
            fcm_se.log(fcm_se.MSG_WARNING, fcm_se.MSG_SENDER_FCM,
"FcmUpnp::uninstall: Unregistration from MessagingSystem failed");
        }

        fcm_se.log(fcm_se.MSG_INIT, fcm_se.MSG_SENDER_FCM, "FcmUpnp:uninstall
finished");
    }

    //
==========================================================================
    //
    //                                  FcmListener
    //                                  ===========
    //  (the receiveMsg method is called when a message for the Fcm arrives. If
it
```

Ⓐ

```
    //   is a standard FcmMessage the according method is called (see below),
which
    //   can be overloaded by the final FCM).
    //
    //
================================================================================
    public class FcmListener
    extends HaviListener
    {
        public FcmListener() throws HaviGeneralException
        {
        }

        public boolean receiveMsg(boolean haveReplied, byte protocolType, SEID
sourceId, SEID destId, Status state, HaviByteArrayInputStream payload)
        {
            boolean i_replied = false;
            if (state.getApiCode() == ConstApiCode.MSG && state.getErrorCode()
== ConstGeneralErrorCode.PRINT_SYSTEM_STATUS) {
                try {
                    fcm_se.log(fcm_se.MSG_INFO_HIGH,
fcm_se.MSG_SENDER_APPLICATION,"==========================================
========================");
                    fcm_se.log(fcm_se.MSG_INFO_HIGH,
fcm_se.MSG_SENDER_APPLICATION,"======= System/Status information FcmUpnp - SEID
"+fcm_se.getSeid());
                    fcm_se.log(fcm_se.MSG_INFO_HIGH,
fcm_se.MSG_SENDER_APPLICATION,"==========================================
========================");
                } catch (HaviGeneralException ex) {
                }
            } else {
                fcm_se.log(fcm_se.MSG_INFO_LOW, fcm_se.MSG_SENDER_FCM, "FcmUpnp
received message");
                if (haveReplied == true) return i_replied;

                // Variables
                StatusHolder rreturn_status = new StatusHolder();
                Status return_status;
                OperationCode received_op_code;
                byte control_flags;
                int transaction_id;

                try {
                    // Check protocol type
                    if (protocolType != ConstProtocolType.HAVI_RMI) return
i_replied;

                    // Check the destination SEID
                    if (destId.equals(fcm_se.getSeid()) == false) return
i_replied;

                    // Read the remaining header
                    payload.reset();
                    received_op_code = new OperationCode(payload);
                    control_flags = payload.readByte();
                    transaction_id = payload.readInt();

                    // Check control flags
                    if (control_flags != 0) return i_replied;

                    // Check API code of opCode
                    if (received_op_code.getApiCode() != ConstApiCode.FCM)
return i_replied;
```

```
                } catch (Exception e) {
                    fcm_se.log(fcm_se.MSG_WARNING, fcm_se.MSG_SENDER_FCM,
"FcmListener: Reading the header failed. Give up!");
                    return i_replied;
                }

                // Ok, looks fine. Now check the operation ID
                try {
                    switch (received_op_code.getOperationId()) {
                        //
                        // GET_HUID
                        // ========
                        //
                        case ConstFcmOperationId.GET_HUID:
                        {
                            fcm_se.log(fcm_se.MSG_INFO_HIGH,
fcm_se.MSG_SENDER_FCM, "FcmUpnp received GetHuid");
                            HUIDHolder huid = new HUIDHolder();
                            return_status = do_GetHuid(huid);
                            fcm_server.getHuidResp(sourceId, return_status,
transaction_id, huid.getValue());
                            i_replied = true;
                        }
                        break;

                        //
                        // GET_DCM_SEID
                        // ============
                        //
                        case ConstFcmOperationId.GET_DCM_SEID:
                        {
                            fcm_se.log(fcm_se.MSG_INFO_HIGH,
fcm_se.MSG_SENDER_FCM, "FcmUpnp received GetDcmSeid");
                            SEIDHolder dcm_seid = new SEIDHolder();
                            return_status = do_GetDcmSeid(dcm_seid);
                            fcm_server.getDcmSeidResp(sourceId, return_status,
transaction_id, dcm_seid.getValue());
                            i_replied = true;
                        }
                        break;

                        //
                        // GET_FCM_TYPE
                        // ============
                        //
                        case ConstFcmOperationId.GET_FCM_TYPE:
                        {
                            fcm_se.log(fcm_se.MSG_INFO_HIGH,
fcm_se.MSG_SENDER_FCM, "FcmUpnp received GetFcmType");
                            IntHolder fcm_type = new IntHolder();
                            return_status = do_GetFcmType(fcm_type);
                            fcm_server.getFcmTypeResp(sourceId, return_status,
transaction_id, fcm_type.getValue());
                            i_replied = true;
                        }
                        break;

                        //
                        // GET_POWER_STATE
                        // ===============
                        //
                        case ConstFcmOperationId.GET_POWER_STATE:
                        {
```

```
                                fcm_se.log(fcm_se.MSG_INFO_HIGH,
fcm_se.MSG_SENDER_FCM, "FcmUpnp received GetPowerState");
                                BooleanHolder get_power_state = new BooleanHolder();
                                return_status = do_GetPowerState(get_power_state);
                                fcm_server.getPowerStateResp(sourceId,
return_status, transaction_id, get_power_state.getValue());
                                i_replied = true;
                            }
                            break;

                            //
                            // SET_POWER_STATE
                            // ===============
                            //
                            case ConstFcmOperationId.SET_POWER_STATE:
                            {
                                fcm_se.log(fcm_se.MSG_INFO_HIGH,
fcm_se.MSG_SENDER_DCM, "FcmUpnp received SetPowerState");
                                BooleanHolder set_power_state = new
BooleanHolder(payload.readBoolean());
                                return_status = do_SetPowerState(set_power_state);
                                fcm_server.setPowerStateResp(sourceId,
return_status, transaction_id, set_power_state.getValue());
                                i_replied = true;
                            }
                            break;

                            //
                            // WINK
                            // ====
                            //
                            case ConstFcmOperationId.WINK:
                            {
                                fcm_se.log(fcm_se.MSG_INFO_HIGH,
fcm_se.MSG_SENDER_DCM, "FcmUpnp received Wink");
                                return_status = do_Wink();
                                fcm_server.winkResp(sourceId, return_status,
transaction_id);
                                i_replied = true;
                            }
                            break;

                            //
                            // UNWINK
                            // ======
                            //
                            case ConstFcmOperationId.UNWINK:
                            {
                                fcm_se.log(fcm_se.MSG_INFO_HIGH,
fcm_se.MSG_SENDER_DCM, "FcmUpnp received Unwink");
                                return_status = do_Unwink();
                                fcm_server.unwinkResp(sourceId, return_status,
transaction_id);
                                i_replied = true;
                            }
                            break;

                            //
                            // CAN_WINK
                            // ========
                            //
                            case ConstFcmOperationId.CAN_WINK:
                            {
```

```
                                fcm_se.log(fcm_se.MSG_INFO_HIGH,
fcm_se.MSG_SENDER_DCM, "FcmUpnp received CanWink");
                                BooleanHolder can_i_wink = new BooleanHolder();
                                return_status = do_CanWink(can_i_wink);
                                fcm_server.canWinkResp(sourceId, return_status,
transaction_id, can_i_wink.getValue());
                                i_replied = true;
                            }
                            break;

                            //
                            // GET_PLUG_COUNT
                            // ==============
                            //
                            case ConstFcmOperationId.GET_PLUG_COUNT:
                            fcm_se.log(fcm_se.MSG_INFO_HIGH, fcm_se.MSG_SENDER_DCM,
"FcmUpnp received GetPlugCount");
                            {
                                ShortHolder in_count  = new ShortHolder();
                                ShortHolder out_count = new ShortHolder();
                                return_status = do_GetPlugCount(in_count,
out_count);
                                fcm_server.getPlugCountResp(sourceId, return_status,
transaction_id, in_count.getValue(), out_count.getValue());
                                i_replied = true;
                            }
                            break;

                            //
                            // GET_SUPPORTED_STREAM_TYPES
                            // ==========================
                            //
                            case ConstFcmOperationId.GET_SUPPORTED_STREAM_TYPES:
                            fcm_se.log(fcm_se.MSG_INFO_HIGH, fcm_se.MSG_SENDER_DCM,
"FcmUpnp received GetSupportedStreamTypes");
                            {
                                short plug_num  = payload.readShort();
                                int   direction = payload.readInt();
                                StreamTypeSeqHolder stream_types = new
StreamTypeSeqHolder();
                                return_status = do_GetSupportedStreamTypes(plug_num,
direction, stream_types);
                                fcm_server.getSupportedStreamTypesResp(sourceId,
return_status, transaction_id, stream_types.getValue());
                                i_replied = true;
                            }
                            break;

                            //
                            // GET_SERVICE_DESCRIPTIONS
                            // ========================
                            //
                            case ConstFcmUpnpOperationId.GET_SERVICE_DESCRIPTIONS:
                            fcm_se.log(fcm_se.MSG_INFO_HIGH, fcm_se.MSG_SENDER_DCM,
"FcmUpnp received GetServiceDescriptions");
                            {
                                IntHolder services = new IntHolder(0);
                                return_status = do_GetServiceDescriptions(services);
                                HaviByteArrayOutputStream buffer = new
HaviByteArrayOutputStream();
                                buffer.writeInt(services.getValue());
                                fcm_se.msgSendResponse(sourceId, received_op_code,
ConstTransferMode.SIMPLE, return_status, buffer, transaction_id);
                                i_replied = true;
```

(B)

```
                    }
                    break;

                    //
                    // GET_DEVICE_TYPE
                    // ===============
                    //
                    case ConstFcmUpnpOperationId.GET_DEVICE_TYPE:
                    fcm_se.log(fcm_se.MSG_INFO_HIGH, fcm_se.MSG_SENDER_DCM,
"FcmUpnp received GetDeviceType");
                    {
                        StringHolder device_type = new StringHolder();
                        return_status = do_GetDeviceType(device_type);
                        HaviByteArrayOutputStream buffer = new
HaviByteArrayOutputStream();
                        buffer.writeHaviString(device_type.getValue());
                        fcm_se.msgSendResponse(sourceId, received_op_code,
ConstTransferMode.SIMPLE, return_status, buffer, transaction_id);
                        i_replied = true;
                    }
                    break;

                    //
                    // GET_SERVICE_INFORMATION_LIST
                    // ============================
                    //
                    case
ConstFcmUpnpOperationId.GET_SERVICE_INFORMATION_LIST:
                    fcm_se.log(fcm_se.MSG_INFO_HIGH, fcm_se.MSG_SENDER_DCM,
"FcmUpnp received GetServiceInformationList");
                    {
                        int iservice = payload.readInt();
                        Vector service_informations = new Vector();
                        return_status =
do_GetServiceInformationList(iservice, service_informations);
                        HaviByteArrayOutputStream buffer = new
HaviByteArrayOutputStream();
                        buffer.writeInt(service_informations.size());
                        if (service_informations.size() > 0) {
                            for (int i=0;i<service_informations.size();i++)
{

((ServiceInformation)service_informations.elementAt(i)).marshal(buffer);
                            }
                        }
                        fcm_se.msgSendResponse(sourceId, received_op_code,
ConstTransferMode.SIMPLE, return_status, buffer, transaction_id);
                        i_replied = true;
                    }
                    break;

                    //
                    // PERFORM_CONTROL_COMMAND
                    // =======================
                    //
                    case ConstFcmUpnpOperationId.PERFORM_CONTROL_COMMAND:
                    fcm_se.log(fcm_se.MSG_INFO_HIGH, fcm_se.MSG_SENDER_DCM,
"FcmUpnp received PerformControlCommand");
                    {
                        String control_url     = payload.readHaviString();
                        String service_type    = payload.readHaviString();
                        String service_type_v  = payload.readHaviString();
                        String action          = payload.readHaviString();
                        int    nvariables      = payload.readInt();
```

```
                                Vector command          = new Vector();
                                Vector variable         = new Vector();
                                Vector return_argument = new Vector();
                                Vector return_value     = new Vector();
                                if (nvariables > 0) {
                                    for (int i=0;i<nvariables;i++) {

command.addElement(payload.readHaviString());

variable.addElement(payload.readHaviString());
                                    }
                                }
                                if (sendControlCommand(new URL(control_url),
service_type, service_type_v, action, command, variable, return_argument,
return_value) == false) {
                                    return_status = new Status(ConstApiCode.FCM,
ConstGeneralErrorCode.UNIDENTIFIED_FAILURE);
                                } else {
                                    return_status = new Status(ConstApiCode.FCM,
ConstGeneralErrorCode.SUCCESS);
                                }
                                HaviByteArrayOutputStream buffer = new
HaviByteArrayOutputStream();
                                if (return_argument.size() != return_value.size()) {
                                    fcm_se.log(fcm_se.MSG_ERROR,
fcm_se.MSG_SENDER_DCM, "FcmUpnp::PERFORM_CONTROL_COMMAND: Mismatch vector
size");
                                    buffer.writeInt(0);
                                } else {
                                    buffer.writeInt(return_argument.size());
                                    if (return_argument.size() > 0) {
                                        String ret_string;
                                        for (int i=0;i<return_argument.size();i++) {
                                            ret_string =
(String)return_argument.elementAt(i);
                                            buffer.writeHaviString(ret_string);
                                            ret_string =
(String)return_value.elementAt(i);
                                            buffer.writeHaviString(ret_string);
                                        }
                                    }
                                }
                                fcm_se.msgSendResponse(sourceId, received_op_code,
ConstTransferMode.SIMPLE, return_status, buffer, transaction_id);
                                i_replied = true;
                            }
                            break;
                            //
                            // PERFORM_DEVICE_VARIABLE_QUERY
                            // ============================
                            //
                            case
ConstFcmUpnpOperationId.PERFORM_DEVICE_VARIABLE_QUERY:
                            fcm_se.log(fcm_se.MSG_INFO_HIGH, fcm_se.MSG_SENDER_DCM,
"FcmUpnp received PerformDeviceVariableQuery");
                            {
                                String       control_url    =
payload.readHaviString();
                                String       variable_name =
payload.readHaviString();
                                StringBuffer return_value  = new StringBuffer();
                                if (queryDeviceVariable(new URL(control_url),
variable_name, return_value) == false) {
```

```
                                return_status = new Status(ConstApiCode.FCM,
ConstGeneralErrorCode.UNIDENTIFIED_FAILURE);
                            } else {
                                return_status = new Status(ConstApiCode.FCM,
ConstGeneralErrorCode.SUCCESS);
                            }
                            HaviByteArrayOutputStream buffer = new
HaviByteArrayOutputStream();
                            buffer.writeHaviString(return_value.toString());
                            fcm_se.msgSendResponse(sourceId, received_op_code,
ConstTransferMode.SIMPLE, return_status, buffer, transaction_id);
                            i_replied = true;
                        }
                        break;

                        //
                        // not yet implemented
                        // ==================
                        //
                        default:
                        // Received unknown command
                        fcm_se.log(fcm_se.MSG_WARNING, fcm_se.MSG_SENDER_FCM,
"FcmUpnp::FcmListener: Received unknown command
("+received_op_code.getOperationId()+"). Send ENOTIMPLEMENTED");
                        fcm_se.msgSendResponse(sourceId, received_op_code,
ConstTransferMode.SIMPLE, new Status(ConstApiCode.FCM,
ConstGeneralErrorCode.NOT_IMPLEMENTED), new
HaviByteArrayOutputStream(),transaction_id);
                        break;
                    }
                } catch (Exception e) {
                    e.printStackTrace();
                    fcm_se.log(fcm_se.MSG_WARNING, fcm_se.MSG_SENDER_FCM,
"FcmUpnp::FcmListener: Sending response failed. Give up!");
                }
            }
            return i_replied;
        }
    }

    //
=============================================================================
    //
    //                  Methods for answering the incoming request.
    //                     (can be overloaded by the final FCM)
    //
    //
=============================================================================

    // --- do_GetHuid ---
    Status do_GetHuid(HUIDHolder huid)
    {
        Status return_status = null;
        try {
            // Default Return code
            return_status = new Status(ConstApiCode.FCM,
ConstGeneralErrorCode.UNIDENTIFIED_FAILURE);
            // Build the response
            huid.setValue(my_huid);
            return_status = new Status(ConstApiCode.FCM,
ConstGeneralErrorCode.SUCCESS);
        } catch (Exception e) {
            fcm_se.log(fcm_se.MSG_WARNING, fcm_se.MSG_SENDER_FCM,
"FcmUpnp::do_GetHuid: Exception occured when building or sending the command");
```

```
        }
        return return_status;
    }

    // --- do_GetDcmSeid ---
    Status do_GetDcmSeid(SEIDHolder seid)
    {
        Status return_status = null;
        try {
            // Default Return code
            return_status = new Status(ConstApiCode.FCM,
ConstGeneralErrorCode.UNIDENTIFIED_FAILURE);
            // Build the response
            seid.setValue(my_dcm_seid);
            return_status = new Status(ConstApiCode.FCM,
ConstGeneralErrorCode.SUCCESS);
        } catch (Exception e) {
            fcm_se.log(fcm_se.MSG_WARNING, fcm_se.MSG_SENDER_FCM,
"FcmUpnp::do_GetDcmSeid: Exception occured when building or sending the
command");
        }
        return return_status;
    }

    // --- do_GetFcmType ---
    Status do_GetFcmType(IntHolder fcm_type)
    {
        Status return_status = null;
        try {
            // Default Return code
            return_status = new Status(ConstApiCode.FCM,
ConstGeneralErrorCode.UNIDENTIFIED_FAILURE);
            // Build the response
            fcm_type.setValue(getSoftwareElementType());
            return_status = new Status(ConstApiCode.FCM,
ConstGeneralErrorCode.SUCCESS);
        } catch (Exception e) {
            fcm_se.log(fcm_se.MSG_WARNING, fcm_se.MSG_SENDER_FCM,
"FcmUpnp::do_GetFcmType: Exception occured when building or sending the
command");
        }
        return return_status;
    }

    // --- do_GetPowerState --- (Must be overloaded by the final Fcm)
    Status do_GetPowerState(BooleanHolder power_state)
    {
        Status return_status = null;
        try {
            // Default Return code
            return_status = new Status(ConstApiCode.FCM,
ConstGeneralErrorCode.UNIDENTIFIED_FAILURE);
            // Build the response
            return_status = new Status(ConstApiCode.FCM,
ConstGeneralErrorCode.NOT_IMPLEMENTED);
        } catch (Exception e) {
            fcm_se.log(fcm_se.MSG_WARNING, fcm_se.MSG_SENDER_FCM,
"FcmUpnp::do_GetPowerState: Exception occured when building or sending the
command");
        }
        return return_status;
    }

    // --- do_SetPowerState --- (Must be overloaded by the final Fcm)
```

```
    Status do_SetPowerState(BooleanHolder power_state)
    {
        Status return_status = null;
        try {
            // Default Return code
            return_status = new Status(ConstApiCode.FCM,
ConstGeneralErrorCode.UNIDENTIFIED_FAILURE);
            // Ignore the requested state. Build the response
            return_status = new Status(ConstApiCode.FCM,
ConstGeneralErrorCode.NOT_IMPLEMENTED);
        } catch (Exception e) {
            fcm_se.log(fcm_se.MSG_WARNING, fcm_se.MSG_SENDER_FCM,
"FcmUpnp::do_SetPowerState: Exception occured when building or sending the
command");
        }
        return return_status;
    }

    // --- do_Wink ---
    Status do_Wink()
    {
        Status return_status = null;
        try {
            // Default Return code
            return_status = new Status(ConstApiCode.FCM,
ConstGeneralErrorCode.NOT_IMPLEMENTED);
        } catch (Exception e) {
            fcm_se.log(fcm_se.MSG_WARNING, fcm_se.MSG_SENDER_FCM,
"FcmUpnp::do_Wink: Exception occured when building or sending the command");
        }
        return return_status;
    }

    // --- do_Unwink ---
    Status do_Unwink()
    {
        Status return_status = null;
        try {
            // Default Return code
            return_status = new Status(ConstApiCode.FCM,
ConstGeneralErrorCode.NOT_IMPLEMENTED);
        } catch (Exception e) {
            fcm_se.log(fcm_se.MSG_WARNING, fcm_se.MSG_SENDER_FCM,
"FcmUpnp::do_Unwink: Exception occured when building or sending the command");
        }
        return return_status;
    }

    // --- do_CanWink ---
    Status do_CanWink(BooleanHolder can_wink)
    {
        Status return_status = null;
        try {
            // Default Return code
            return_status = new Status(ConstApiCode.FCM,
ConstGeneralErrorCode.UNIDENTIFIED_FAILURE);
            // Build the response
            can_wink.setValue(false);
            return_status = new Status(ConstApiCode.FCM,
ConstGeneralErrorCode.SUCCESS);
        } catch (Exception e) {
            fcm_se.log(fcm_se.MSG_WARNING, fcm_se.MSG_SENDER_FCM,
"FcmUpnp::do_CanWink: Exception occured when building or sending the command");
        }
```

```
        return return_status;
    }

    // --- do_GetPlugCount (Not really implemented - must be done by the final
FCM) ---
    Status do_GetPlugCount(ShortHolder in_count, ShortHolder out_count)
    {
        Status return_status = null;
        try {
            in_count.setValue((short)0);
            out_count.setValue((short)0);
            // Return NOT IMPLEMENTED
            return_status = new Status(ConstApiCode.FCM,
ConstGeneralErrorCode.NOT_IMPLEMENTED);
        } catch (Exception e) {
            fcm_se.log(fcm_se.MSG_WARNING, fcm_se.MSG_SENDER_FCM,
"FcmUpnp::do_GetPlugCount: Exception occured when performing the request");
        }
        return return_status;
    }

    // --- do_GetSupportedStreamTypes (Not really implemented - must be done by
the final FCM) ---
    Status do_GetSupportedStreamTypes(short plug_num, int direction,
StreamTypeSeqHolder stream_types)
    {
        Status return_status = null;
        try {
            StreamType[] st = new StreamType[0];
            stream_types.setValue(st);
            // Return NOT IMPLEMENTED
            return_status = new Status(ConstApiCode.FCM,
ConstGeneralErrorCode.NOT_IMPLEMENTED);
        } catch (Exception e) {
            fcm_se.log(fcm_se.MSG_WARNING, fcm_se.MSG_SENDER_FCM,
"FcmUpnp::do_GetSupportedStreamTypes: Exception occured when performing the
request");
        }
        return return_status;
    }

    // --- do_GetServiceDescriptions ---
    Status do_GetServiceDescriptions(IntHolder services)
    {
        Status return_status = null;
        try {
            services.setValue(service_list.size());
            return_status = new Status(ConstApiCode.FCM,
ConstGeneralErrorCode.SUCCESS);
        } catch (Exception e) {
            fcm_se.log(fcm_se.MSG_WARNING, fcm_se.MSG_SENDER_FCM,
"FcmUpnp::do_GetServiceDescriptions: Exception occured when performing the
request");
        }
        return return_status;
    }

    // --- do_GetDeviceType ---
    Status do_GetDeviceType(StringHolder device_type)
    {
        Status return_status = null;
        try {
            device_type.setValue(my_device_type);
```

```
            return_status = new Status(ConstApiCode.FCM,
ConstGeneralErrorCode.SUCCESS);
        } catch (Exception e) {
            fcm_se.log(fcm_se.MSG_WARNING, fcm_se.MSG_SENDER_FCM,
"FcmUpnp::do_GetDeviceType: Exception occured when performing the request");
        }
        return return_status;
    }

    // --- do_GetServiceInformationList ---
    Status do_GetServiceInformationList(int iservice, Vector
service_informations)
    {
        Status return_status = null;
        try {
            if (iservice+1 <= service_list.size()) {
                ServiceInformation[] service_information =
((ServiceDescription)service_list.elementAt(iservice)).getServiceInformationList
();
                if (service_information.length > 0) {
                    for (int i=0;i<service_information.length;i++) {
                        service_informations.add(service_information[i]);
                    }
                }
            }
            return_status = new Status(ConstApiCode.FCM,
ConstGeneralErrorCode.SUCCESS);
        } catch (Exception e) {
            fcm_se.log(fcm_se.MSG_WARNING, fcm_se.MSG_SENDER_FCM,
"FcmUpnp::do_GetServiceInformationList: Exception occured when performing the
request");
        }
        return return_status;
    }


    //
==========================================================================
    //
    //                                  Additional Methods
    //                                  ==================
    //
    //
==========================================================================

    //
    // do_the_registration
    // ===================
    // This method does the registration at the Registry. It is assumed that all
FCMs
    // register the same elements. But they differ in the ATT_SE_TYPE, so this
attribute
    // is created by a special method which should be overloaded by the final
FCM.
    void do_the_registration()
    {
        Vector attribute_list = new Vector(); // Contains all attributes
        HaviByteArrayOutputStream hbaos;
        org.havi.types.Attribute attribute;
        SEID my_seid;

        // Get own SEID
        try {
            my_seid = fcm_se.getSeid();
```

```
        } catch (HaviGeneralException e) {
            fcm_se.log(fcm_se.MSG_ERROR, fcm_se.MSG_SENDER_FCM,
"FcmUpnp::do_the_registration: Fetching SEID failed");
            return;
        }

        // --- Build all attributes ---
        // SoftwareElementType
        try {
            hbaos = new HaviByteArrayOutputStream();
            hbaos.writeInt(getSoftwareElementType());
            attribute = new
org.havi.types.Attribute(ConstAttributeName.ATT_SE_TYPE, hbaos);
            attribute_list.addElement(attribute.clone());
        } catch (Exception e) {
            fcm_se.log(fcm_se.MSG_WARNING, fcm_se.MSG_SENDER_FCM,
"FcmUpnp::do_the_registration: Error when creating the SoftwareElementType");
        }

        // Vendor ID
        try {
            hbaos = new HaviByteArrayOutputStream();
            my_vendor_id.marshal(hbaos);
            attribute = new
org.havi.types.Attribute(ConstAttributeName.ATT_VENDOR_ID, hbaos);
            attribute_list.addElement(attribute.clone());
        } catch (Exception e) {
            fcm_se.log(fcm_se.MSG_WARNING, fcm_se.MSG_SENDER_FCM,
"FcmUpnp::do_the_registration: Error when creating the VendorId");
        }

        // HUID
        try {
            hbaos = new HaviByteArrayOutputStream();
            my_huid.marshal(hbaos);
            attribute = new
org.havi.types.Attribute(ConstAttributeName.ATT_HUID, hbaos);
            attribute_list.addElement(attribute.clone());
        } catch (Exception e) {
            fcm_se.log(fcm_se.MSG_WARNING, fcm_se.MSG_SENDER_FCM,
"FcmUpnp::do_the_registration: Error when creating the HUID");
        }

        // TargetId - not yet implemented
        try {
            hbaos = new HaviByteArrayOutputStream();
            my_target_id.marshal(hbaos);
            attribute = new
org.havi.types.Attribute(ConstAttributeName.ATT_TARGET_ID, hbaos);
            attribute_list.addElement(attribute.clone());
        } catch (Exception e) {
            fcm_se.log(fcm_se.MSG_WARNING, fcm_se.MSG_SENDER_FCM,
"FcmUpnp::do_the_registration: Error when creating the TargetId");
        }

        // InterfaceId - not yet implemented
        try {
            hbaos = new HaviByteArrayOutputStream();
            hbaos.writeShort((int)my_interface_id);
            attribute = new
org.havi.types.Attribute(ConstAttributeName.ATT_INTERFACE_ID, hbaos);
            attribute_list.addElement(attribute.clone());
        } catch (Exception e) {
```

```
            fcm_se.log(fcm_se.MSG_WARNING, fcm_se.MSG_SENDER_FCM,
"FcmUpnp::do_the_registration: Error when creating the InterfaceId");
        }

        // DeviceClass
        try {
            hbaos = new HaviByteArrayOutputStream();
            hbaos.writeInt(my_device_class);
            attribute = new
org.havi.types.Attribute(ConstAttributeName.ATT_DEVICE_CLASS, hbaos);
            attribute_list.addElement(attribute.clone());
        } catch (Exception e) {
            fcm_se.log(fcm_se.MSG_WARNING, fcm_se.MSG_SENDER_FCM,
"FcmUpnp::do_the_registration: Error when creating the DeviceClass");
        }

        // DeviceManufacturer
        try {
            hbaos = new HaviByteArrayOutputStream();
            hbaos.writeHaviString(my_device_manufacturer);
            attribute = new
org.havi.types.Attribute(ConstAttributeName.ATT_DEVICE_MANUF, hbaos);
            attribute_list.addElement(attribute.clone());
        } catch (Exception e) {
            fcm_se.log(fcm_se.MSG_WARNING, fcm_se.MSG_SENDER_FCM,
"FcmUpnp::do_the_registration: Error when creating the DeviceManufacturer");
        }

        // SoftwareElementVersion
        try {
            hbaos = new HaviByteArrayOutputStream(4);
            byte[] se_version = new byte[4];
            se_version[0]=0;
            se_version[1]=1;
            se_version[2]=0;
            se_version[3]=0;
            hbaos.write(se_version);
            attribute = new
org.havi.types.Attribute(ConstAttributeName.ATT_SE_VERS, hbaos);
            attribute_list.addElement(attribute.clone());
        } catch (Exception e) {
            fcm_se.log(fcm_se.MSG_WARNING, fcm_se.MSG_SENDER_FCM,
"FcmUpnp::do_the_registration: Error when creating the SoftwareElementVersion");
        }

        // UserPreferredName
        try {
            hbaos = new HaviByteArrayOutputStream();
            hbaos.writeHaviString(my_user_preferred_name);
            attribute = new
org.havi.types.Attribute(ConstAttributeName.ATT_USER_PREF_NAME, hbaos);
            attribute_list.addElement(attribute.clone());
        } catch (Exception e) {
            fcm_se.log(fcm_se.MSG_WARNING, fcm_se.MSG_SENDER_FCM,
"FcmUpnp::do_the_registration: Error when creating the UserPreferredName");
        }

        if (getDdiCapability() == true) {
            try {
                hbaos = new HaviByteArrayOutputStream();
                hbaos.writeInt(ConstGuiReq.DDI_GUI);
                attribute = new
org.havi.types.Attribute(ConstAttributeName.ATT_GUI_REQ, hbaos);
                attribute_list.addElement(attribute.clone());
```

```
            } catch (Exception e) {
                fcm_se.log(fcm_se.MSG_WARNING, fcm_se.MSG_SENDER_FCM,
"FcmUpnp::do_the_registration: Error when creating the GuiReq");
            }
        }

        // Build the AttributeList
        org.havi.types.Attribute attributes[] = new
org.havi.types.Attribute[attribute_list.size()];
        for (int i=0;i<attribute_list.size();i++) {
            attributes[i]=(org.havi.types.Attribute)attribute_list.elementAt(i);
        }

        // Do the registration
        try {
            RegistryLocalClient reg = new RegistryLocalClient(fcm_se);
            reg.registerElementSync(10000, my_seid,attributes);
        } catch (Exception e) {
            fcm_se.log(fcm_se.MSG_WARNING, fcm_se.MSG_SENDER_FCM,
"FcmUpnp::do_the_registration: Registration failed");
        }
    }

    //
    // getSoftwareElementType
    // ======================
    // This method provides the SoftwareElementType for the methods
    // do_the_registration() and get_fcm_type_received().
    // This method should be overwritten by the final FCM!
    //
    int getSoftwareElementType()
    {
        return ConstSoftwareElementType.GENERIC_FCM;
    }


    //
    // getDdiCapability
    // ================
    // This methods returns true if the FCM is DDI capable or false if it is
not.
    // The method shall me overwritten by the final FCM.
    boolean getDdiCapability()
    {
        return false;
    }

    //
    // sendControlCommand
    // ==================
    // A UPnP control command is sent to an UPnP device.
    boolean sendControlCommand(URL url, String service_type, String
service_type_v, String action_name, Vector in_argument_names, Vector
in_argument_values, Vector out_argument_names, Vector out_argument_values)
    {
        boolean ret_value = true;

        try {
            // Build the body of the SOAP Message
            ByteArrayOutputStream baos_body = new ByteArrayOutputStream();
            baos_body.write((new String("<s:Envelope\r\n")).getBytes());
            baos_body.write((new String("
xmlns:s=\"http://schemas.xmlsoap.org/soap/envelope/\"\r\n")).getBytes());
```

```
            baos_body.write((new String("
s:encodingStyle=\"http://schemas.xmlsoap.org/soap/encoding/\">\r\n")).getBytes()
);
            baos_body.write((new String("  <s:Body>\r\n")).getBytes());
            baos_body.write((new String("    <u:"+action_name+"
xmlns:u=\"urn:schemas-upnp-
org:service:"+service_type+":"+service_type_v+"\">\r\n")).getBytes());
            if (in_argument_names.size() != 0) {
                if (in_argument_names.size() != in_argument_values.size()) {
                    fcm_se.log(fcm_se.MSG_WARNING, fcm_se.MSG_SENDER_FCM,
"FcmUpnp::sendControlCommand: Mismatch in vector lenghth of in_argument_names
and in_argument_values!");
                } else {
                    for (int i=0;i<in_argument_names.size();i++) {
                        baos_body.write((new String("
<"+(String)in_argument_names.elementAt(i)+">"+(String)in_argument_values.element
At(i)+"</"+(String)in_argument_names.elementAt(i)+">\r\n")).getBytes());
                    }
                }
            }
            baos_body.write((new String("
</u:"+action_name+">\r\n")).getBytes());
            baos_body.write((new String("  </s:Body>\r\n")).getBytes());
            baos_body.write((new String("</s:Envelope>\r\n")).getBytes());

            // Build the header of the SOAP message
            ByteArrayOutputStream baos_header = new ByteArrayOutputStream();
            baos_header.write((new String("POST "+url.getFile()+"
HTTP/1.1\r\n")).getBytes());
            baos_header.write((new String("HOST:
"+url.getHost()+":"+url.getPort()+"\r\n")).getBytes());
            baos_header.write((new String("CONTENT-LENGTH:
"+baos_body.toByteArray().length+"\r\n")).getBytes());
            baos_header.write((new String("CONTENT-TYPE: text/xml;
charset=\"utf-8\"\r\n")).getBytes());
            baos_header.write((new String("SOAPACTION: \"urn:schemas-upnp-
org:service:"+service_type+":"+service_type_v+"#"+action_name+"\"\r\n")).getByte
s());
            baos_header.write((new String("\r\n")).getBytes());
            //System.out.println(baos_header.toString());
            //System.out.println(baos_body.toString());

            // Create the Socket for the connection to the device
            Socket sock = new Socket(url.getHost(), url.getPort());
            sock.setSoTimeout(30000);
            OutputStream out = sock.getOutputStream();
            InputStream in = sock.getInputStream();

            // Send the Command
            out.write(baos_header.toByteArray());
            out.write(baos_body.toByteArray());

            // Read the Response
            try {
                int len;
                byte b[] = new byte[100];
                StringBuffer sb = new StringBuffer();
                while ((len = in.read(b)) != -1) {
                    sb.append(new String(b,0,len));
                }

                StringTokenizer st = new StringTokenizer(sb.toString(), "\n");
                // Check for correct header
                if (st.hasMoreElements() == true) {
```

```
                    String header_line_1 = st.nextToken().trim();
                    if (header_line_1.equals("HTTP/1.1 200 OK") == false) {
                        fcm_se.log(fcm_se.MSG_WARNING, fcm_se.MSG_SENDER_FCM,
"FcmUpnp::sendControlCommand: Received wrong header in response from the UPnP
device: "+header_line_1);
                        ret_value = false;
                    } else {
                        // Skip the rest header
                        while (st.hasMoreElements() == true) {
                            if (st.nextToken().trim().length() == 0) break;
                        }
                        // Read the body
                        StringBuffer sbe = new StringBuffer();
                        while (st.hasMoreElements() == true) {
                            sbe.append(st.nextToken().trim()+"\n");
                        }
                        //System.out.println("Response \n"+sbe.toString());

                        // Create an XML element for the body
                        SAXBuilder builder = new SAXBuilder();
                        Document doc = builder.build(new
ByteArrayInputStream(sbe.toString().getBytes()));

                        // Fetch root element and Namespace
                        Element root = doc.getRootElement();
                        if (root == null) {
                            fcm_se.log(fcm_se.MSG_WARNING,
fcm_se.MSG_SENDER_FCM, "FcmUpnp::sendControlCommand: Didn't find root
element!");
                            throw new
HaviUnidentifiedFailureException(ConstApiCode.FCM);
                        }
                        Namespace ns = root.getNamespace();

                        // Fetch the body element
                        Element body = root.getChild("Body",ns);
                        if (root == null) {
                            fcm_se.log(fcm_se.MSG_WARNING,
fcm_se.MSG_SENDER_FCM, "FcmUpnp::sendControlCommand: Didn't find body
element!");
                            throw new
HaviUnidentifiedFailureException(ConstApiCode.FCM);
                        }

                        // Fetch response
                        List response_list = body.getChildren();
                        if (response_list.size() != 1) {
                            fcm_se.log(fcm_se.MSG_WARNING,
fcm_se.MSG_SENDER_FCM, "FcmUpnp::sendControlCommand: Wrong number of children in
body!");
                            throw new
HaviUnidentifiedFailureException(ConstApiCode.FCM);
                        }
                        Element response = (Element)response_list.get(0);
                        if (response.getName().equals(action_name+"Response") ==
false) {
                            fcm_se.log(fcm_se.MSG_WARNING,
fcm_se.MSG_SENDER_FCM, "FcmUpnp::sendControlCommand: Didn't find child with name
"+action_name+"Response!");
                            throw new
HaviUnidentifiedFailureException(ConstApiCode.FCM);
                        }

                        // Fetch the Out-Parameters
```

```
                        List parameter_list = response.getChildren();
                        if (parameter_list.size() > 0) {
                            for (int i=0;i<parameter_list.size();i++) {
                                Element parameter =
(Element)parameter_list.get(i);

out_argument_names.addElement(parameter.getName());

out_argument_values.addElement(parameter.getText());
                            }
                        }
                    }
                } catch (java.io.IOException ex) {
                    fcm_se.log(fcm_se.MSG_WARNING, fcm_se.MSG_SENDER_FCM,
"FcmUpnp::sendControlCommand: Didn't receive a response from the UPnP
device!!!");
                    ret_value = false;
                } catch (org.jdom.JDOMException e) {
                    e.printStackTrace();
                    fcm_se.log(fcm_se.MSG_WARNING, fcm_se.MSG_SENDER_FCM,
"FcmUpnp::sendControlCommand: JDOMException!");
                    ret_value = false;
                } catch (HaviUnidentifiedFailureException ex) {
                    ret_value = false;
                }

                // Close streams and socket
                in.close();
                out.close();
                sock.close();

            } catch (java.net.UnknownHostException e) {
                fcm_se.log(fcm_se.MSG_WARNING, fcm_se.MSG_SENDER_FCM,
"FcmUpnp::sendControlCommand: Host is unknown!");
                ret_value = false;
            } catch (java.net.SocketException e) {
                fcm_se.log(fcm_se.MSG_WARNING, fcm_se.MSG_SENDER_FCM,
"FcmUpnp::sendControlCommand: SocketException!");
                ret_value = false;
            } catch (java.io.IOException ex) {
                fcm_se.log(fcm_se.MSG_WARNING, fcm_se.MSG_SENDER_FCM,
"FcmUpnp::sendControlCommand: IOException!");
                ret_value = false;
            }

            return ret_value;
        }

        boolean sendControlCommand(URL url, String service_type, String
service_type_v, String action_name, String argument_name, String argument_value)
        {
            Vector argument_names  = new Vector();
            Vector argument_values = new Vector();
            argument_names.addElement(argument_name);
            argument_values.addElement(argument_value);
            return sendControlCommand(url, service_type, service_type_v,
action_name, argument_names, argument_values, new Vector(), new Vector());
        }

        boolean sendControlCommand(URL url, String service_type, String
service_type_v, String action_name)
        {
```

```
        return sendControlCommand(url, service_type, service_type_v,
action_name, new Vector(), new Vector(), new Vector(), new Vector());
    }

    //
    // queryDeviceVariable
    // ===================
    // A variable of an UPnP device is queried.
    boolean queryDeviceVariable(URL url, String variable_name, StringBuffer
return_value)
    {
        boolean ret_value = true;

        try {
            // Build the body of the SOAP Message
            ByteArrayOutputStream baos_body = new ByteArrayOutputStream();
            baos_body.write((new String("<s:Envelope\r\r")).getBytes());
            baos_body.write((new String("
xmlns:s=\"http://schemas.xmlsoap.org/soap/envelope/\"\r\n")).getBytes());
            baos_body.write((new String("
s:encodingStyle=\"http://schemas.xmlsoap.org/soap/encoding/\">\r\n")).getBytes()
);
            baos_body.write((new String("  <s:Body>\r\n")).getBytes());
            baos_body.write((new String("    <u:QueryStateVariable
xmlns:u=\"urn:schemas-upnp-org:control-1-0\">\r\n")).getBytes());
            baos_body.write((new String("
<u:varName>"+variable_name+"</u:varName>\r\n")).getBytes());
            baos_body.write((new String("
</u:QueryStateVariable>\r\n")).getBytes());
            baos_body.write((new String("  </s:Body>\r\r")).getBytes());
            baos_body.write((new String("</s:Envelope>\r\n")).getBytes());

            // Build the header of the SOAP message
            ByteArrayOutputStream baos_header = new ByteArrayOutputStream();
            baos_header.write((new String("POST "+url.getFile()+"
HTTP/1.1\r\n")).getBytes());
            baos_header.write((new String("HOST:
"+url.getHost()+":"+url.getPort()+"\r\n")).getBytes());
            baos_header.write((new String("CONTENT-LENGTH:
"+baos_body.toByteArray().length+"\r\n")).getBytes());
            baos_header.write((new String("CONTENT-TYPE: text/xml;
charset=\"utf-8\"\r\n")).getBytes());
            baos_header.write((new String("SOAPACTION: \"urn:schemas-upnp-
org:control-1-0#QueryStateVariable\"\r\n")).getBytes());
            baos_header.write((new String("\r\n")).getBytes());
            //System.out.println(baos_header.toString());
            //System.out.println(baos_body.toString());

            // Create the Socket for the connection to the device
            Socket sock = new Socket(url.getHost(), url.getPort());
            sock.setSoTimeout(30000);
            OutputStream out = sock.getOutputStream();
            InputStream in = sock.getInputStream();

            // Send the Command
            out.write(baos_header.toByteArray());
            out.write(baos_body.toByteArray());

            // Read the Response
            try {
                int len;
                byte b[] = new byte[100];
                StringBuffer sb = new StringBuffer();
                while ((len = in.read(b)) != -1) {
```

```
                    sb.append(new String(b,0,len));
                }
                //System.out.println("Response:\n"+sb);

                StringTokenizer st = new StringTokenizer(sb.toString(), "\n");
                // Check for correct header
                if (st.hasMoreElements() == true) {
                    String header_line_1 = st.nextToken().trim();
                    if (header_line_1.equals("HTTP/1.1 200 OK") == false) {
                        fcm_se.log(fcm_se.MSG_WARNING, fcm_se.MSG_SENDER_FCM,
"FcmUpnp::queryDeviceVariable: Received wrong header in response from the UPnP
device: "+header_line_1);
                        ret_value = false;
                    } else {
                        // Skip the rest header
                        while (st.hasMoreElements() == true) {
                            if (st.nextToken().trim().length() == 0) break;
                        }
                        // Read the body
                        StringBuffer sbe = new StringBuffer();
                        while (st.hasMoreElements() == true) {
                            sbe.append(st.nextToken().trim()+"\n");
                        }

                        // Create an XML element for the body
                        SAXBuilder builder = new SAXBuilder();
                        Document doc = builder.build(new
ByteArrayInputStream(sbe.toString().getBytes()));

                        // Fetch root element and Namespace
                        Element root = doc.getRootElement();
                        if (root == null) {
                            fcm_se.log(fcm_se.MSG_WARNING,
fcm_se.MSG_SENDER_FCM, "FcmUpnp::queryDeviceVariable: Didn't find root
element!");
                            throw new
HaviUnidentifiedFailureException(ConstApiCode.FCM);
                        }
                        Namespace ns = root.getNamespace();

                        // Fetch the body element
                        Element body = root.getChild("Body",ns);
                        if (body == null) {
                            fcm_se.log(fcm_se.MSG_WARNING,
fcm_se.MSG_SENDER_FCM, "FcmUpnp::queryDeviceVariable: Didn't find body
element!");
                            throw new
HaviUnidentifiedFailureException(ConstApiCode.FCM);
                        }

                        // Fetch the response
                        List response_list = body.getChildren();
                        if (response_list.size() != 1) {
                            fcm_se.log(fcm_se.MSG_WARNING,
fcm_se.MSG_SENDER_FCM, "FcmUpnp::queryDeviceVariable: Wrong number of children
in body!");
                            throw new
HaviUnidentifiedFailureException(ConstApiCode.FCM);
                        }
                        Element response = (Element)response_list.get(0);
                        if
(response.getName().equals("QueryStateVariableResponse") == false) {
```

```
                                fcm_se.log(fcm_se.MSG_WARNING,
fcm_se.MSG_SENDER_FCM, "FcmUpnp::queryDeviceVariable: Didn't find
QueryStateVariableResponse element!");
                                throw new
HaviUnidentifiedFailureException(ConstApiCode.FCM);
                            }

                            // Fetch the return value
                            List return_list = response.getChildren();
                            if (return_list.size() != 1) {
                                fcm_se.log(fcm_se.MSG_WARNING,
fcm_se.MSG_SENDER_FCM, "FcmUpnp::queryDeviceVariable: Wrong number of children
in body!");
                                throw new
HaviUnidentifiedFailureException(ConstApiCode.FCM);
                            }
                            Element returnv = (Element)return_list.get(0);
                            if (returnv.getName().equals("return") == false) {
                                fcm_se.log(fcm_se.MSG_WARNING,
fcm_se.MSG_SENDER_FCM, "FcmUpnp::queryDeviceVariable: Didn't find child with
name \"return\"!");
                                    throw new
HaviUnidentifiedFailureException(ConstApiCode.FCM);
                                }

                                // Fetch the Out-Parameters
                                return_value.append(returnv.getText());
                            }
                        }
                    } catch (java.io.IOException ex) {
                        fcm_se.log(fcm_se.MSG_WARNING, fcm_se.MSG_SENDER_FCM,
"FcmUpnp::queryDeviceVariable:: Didn't receive a response from the UPnP
device!!!");
                        ret_value = false;
                    } catch (org.jdom.JDOMException e) {
                        e.printStackTrace();
                        fcm_se.log(fcm_se.MSG_WARNING, fcm_se.MSG_SENDER_FCM,
"FcmUpnp::queryDeviceVariable:: JDOMException!");
                        ret_value = false;
                    } catch (HaviUnidentifiedFailureException ex) {
                        ret_value = false;
                    }

                    // Close streams and socket
                    in.close();
                    out.close();
                    sock.close();
                } catch (java.net.UnknownHostException e) {
                    fcm_se.log(fcm_se.MSG_WARNING, fcm_se.MSG_SENDER_FCM,
"FcmUpnp::queryDeviceVariable:: Host is unknown!");
                    ret_value = false;
                } catch (java.net.SocketException e) {
                    fcm_se.log(fcm_se.MSG_WARNING, fcm_se.MSG_SENDER_FCM,
"FcmUpnp::queryDeviceVariable:: SocketException!");
                    ret_value = false;
                } catch (java.io.IOException ex) {
                    fcm_se.log(fcm_se.MSG_WARNING, fcm_se.MSG_SENDER_FCM,
"FcmUpnp::queryDeviceVariable:: IOException!");
                    ret_value = false;
                }

                return ret_value;
            }
```

```
    //
    // receiveHttpNotifyData
    // =====================
    //
    // This method is called by the UpnpWebServer when an event notification is
received from the
    // device where this FCM subscribed for EventNotification.
    //
    public void receiveHttpNotifyData(String[] header, InputStream data,
OutputStream response)
    {
        // Variables
        int      content_length = -1;
        String   http_response_header_success = new String("HTTP/1.1 200 OK");
        String   http_response_header         = new
String(http_response_header_success);

        // Read the Header first
        try {
            //for (int i=0;i<header.length;i++)
System.out.println("FcmUpnp:receiveHttpNotifyData received header: "+header[i]);
            // Search for the content length
            for (int i=0;i<header.length;i++) {
                StringTokenizer st = new StringTokenizer(header[i],":");
                if (st.hasMoreTokens() == true) {
                    if (st.nextToken().trim().equalsIgnoreCase("CONTENT-LENGTH")
== true) {
                        content_length =
Integer.parseInt(st.nextToken().trim());
                    }
                }
            }

            if (content_length == -1) {
                // The header doesn't contain the CONTENT LENGTH
                http_response_header = new String("HTTP/1.1 412 Missing CONTENT-
LENGTH");
            }

            if (http_response_header.equals(http_response_header_success) ==
true) {
                // Read the XML description of the action
                byte[] bdata = new byte[content_length];
                data.read(bdata);
                {
                    // Overwrite added zeros with LFs. Evidently bug in AXIS
implementation!
                    for (int pos=bdata.length-1;bdata[pos] == 0;pos--)
bdata[pos]=0x0A;
                }
                HaviByteArrayInputStream bais = new
HaviByteArrayInputStream(bdata);
                SAXBuilder builder = new SAXBuilder();
                Document doc = builder.build(bais);

                // Fetch the root element
                Element root = doc.getRootElement();
                if (root == null) {
                    SoftwareElement.log(SoftwareElement.MSG_WARNING,
"FcmUpnp.receiveHttpNotifyData(): No root element found! Give up!");
                    throw new HaviUnidentifiedFailureException((short)0);
                }
```

```
                // Check the namespace
                if (root.getNamespace().getURI().equalsIgnoreCase("urn:schemas-
upnp-org:event-1-0") == false) {
                    SoftwareElement.log(SoftwareElement.MSG_WARNING,
"FcmUpnp.receiveHttpNotifyData(): Wrong namespace:
>"+root.getNamespace().getURI()+"< instead of >urn:schemas-upnp-org:event-1-0<!
Give up!");
                    throw new HaviUnidentifiedFailureException((short)0);
                }

                // Fetch all propertys
                java.util.List propertys = root.getChildren();
                for (int j=0;j<propertys.size();j++) {
                    Element   property_element = (Element)propertys.get(j);
                    // Fetch the variable
                    java.util.List variables = property_element.getChildren();
                    for (int jj=0;jj<variables.size();jj++) {
                        Element   variable_element = (Element)variables.get(jj);
                        System.out.println("Variable ----------
"+variable_element.getName()+"---"+variable_element.getText());
                    }

                }
            }
        } catch (Exception ex) {
            SoftwareElement.log(SoftwareElement.MSG_WARNING,
"FcmUpnp.receiveHttpNotifyData(): Exception occured when intepreting the
data!");
            ex.printStackTrace();
            http_response_header = new String("HTTP/1.1 500 Internal Server
Error");
        }

        // Build the complete response.
        try {
            // Build the header of the SOAP message
            ByteArrayOutputStream baos_header = new ByteArrayOutputStream();
            baos_header.write((new
String(http_response_header+"\n\r")).getBytes());
            baos_header.write((new String("\r\n")).getBytes());
            response.write(baos_header.toByteArray());
        } catch (java.io.IOException ex) {
            SoftwareElement.log(SoftwareElement.MSG_WARNING,
"FcmUpnp.receiveHttpNotifyData(): Exception occured when building the
response!");
            ex.printStackTrace();
        }
    }

    //
============================================================================
    //
    //                                Additional Methods (possibly overloaded)
    //                                ========================
    //
    //
============================================================================

    //
    // getDevicePath
    // =============
    //
    // The root device path for storing information on this device at the
webserver is returned.
```

```
// This method must be overloaded.
//
String getDevicePath()
{
    try {
        StringBuffer sb=new StringBuffer();
        byte[] seid = fcm_se.getSeid().getValue();
        for (int i=0;i<seid.length;i++) {
            int v=(int)seid[i];
            if (v < 0) v+=256;
            String hstring = Integer.toHexString(v);
            if (hstring.length() == 2) {
                sb.append(hstring);
            } else {
                sb.append("0"+hstring);
            }
        }
        return "/upnp/"+sb.toString().toUpperCase();
    } catch (Exception ex) {
        return "/upnp/undefinedguid";
    }
}

//
// getUpnpEventNotificationPath
// ============================
//
// The path to which event notifications are sent by the according UPnP
device.
//
String getUpnpEventNotificationPath()
{
    return getDevicePath()+"/event_notification";
}
```

Fig. 4

```
//-------------------------------------------------------------------------
// Copyright (c) 2000-2001, This software is the property of Thomson
// multimedia, and shall not be reproduced, copied, or distributed
// without written permission.
//-------------------------------------------------------------------------

//-------------------------------------------------------------------------
// TITLE:         HavletUpnpDevice - Implementation of the HavletUpnpDevice
class.
// AUTHOR(s):     Ingo Huetter
// DATE CREATED: 28. September 2001
// FILENAME:      HavletUpnpDevice.java
// PROJECT:       DHN (HAVi implementation)
// COPYRIGHT:     (C) 2000-2001 THOMSON multimedia,
//  THIS SOFTWARE IS THE PROPERTY OF THOMSON multimedia, AND SHALL NOT
//  BE REPRODUCED, COPIED, OR DISTRIBUTED WITHOUT WRITTEN PERMISSION.
//
// VERSION: 1.1
//
// MODIFICATION HISTORY:
//    DATE MODIFIED:
//    AUTHOR(s):
//    MODIFICATION MADE:
//    PROBLEM CAUSING MODIFICATION:
//
// DESCRIPTION:
//  The HavletUpnpDevice creates a graphical UI based on the XML description of
the device.
//  Information on services are retrieved from the FCM using a proprietary API.
//
// NEEDED RESOURCES:
//  The following classes and images are needed by this havlet, so the
//  according files must be packed into the jar file.
//   - com.thmulti.havi.havlet.HavletBitmapComponent
//   - com.thmulti.havi.havlet.HavletResource
//   - com.thmulti.havi.util.BackgroundMosaicWindow
//   - com.thmulti.havi.util.TPanel
//   - com.thmulti.havi.upnp.HavletUpnpDevice
//   - com.thmulti.havi.upnp.ConstFcmUpnpOperationId
//   - com.thmulti.havi.upnp.ServiceInformation
//   - com.thmulti.havi.upnp.upnp_logo.gif
//-------------------------------------------------------------------------

package com.thmulti.havi.upnp;

import org.havi.constants.*;
import org.havi.types.*;
import org.havi.system.*;
import java.awt.*;
import java.awt.event.*;
import java.lang.reflect.*;
import java.util.Vector;
import com.thmulti.havi.util.BackgroundMosaicWindow;
import com.thmulti.havi.util.TPanel;
import com.thmulti.havi.havlet.HavletBitmapComponent;
import com.thmulti.havi.havlet.HavletResource;

public class HavletUpnpDevice
{
    // --- Variables ---
    HavletListener          hv_listener;
    SoftwareElement         hv_se;
    OperationCode           hv_event_opcode;
    UninstallationListener  hv_ul;
```

```
    EventManagerLocalClient em;
    DcmClient               dcm_client;
    Window                  win = null;
    SEID                    dcm_seid;
    SEID                    fcm_seid;
    SEIDSeqHolder           fcm_seids;
    int                     window_height;
    int                     window_width;
    int                     private_log_method_available = -1; // -1 unknown, 0
no, 1 yes
    Method                  private_log_method = null;
    Panel                   sp_panel;
    HavletThread            havlet_thread;

    // --- Constructor ---
    public HavletUpnpDevice()
    {
    }

    // --- install ---
    public int install(SEID source, UninstallationListener listener)
    {
        // Save the parameters
        hv_ul    = listener;
        dcm_seid = source;

        // Create the listener and the software element
        try {
            hv_listener = new HavletListener();
            hv_se       = new SoftwareElement(hv_listener);
        } catch (Exception e) {
            log("ERROR", "HavletUpnpDevice::install: Creating hv_listener or
hv_se failed!");
            return 1;
        }

        // Retrieve the SEID of the FCM(s)
        try {
              dcm_client = new DcmClient(hv_se, dcm_seid);
            fcm_seids  = new SEIDSeqHolder();
              dcm_client.getFcmSeidListSync(3000, fcm_seids);
        } catch (Exception e) {
            log("WARNING", "HavletUpnpDevice::install: Exception when retrieving
the list of FCMs in DCM");
            return 1;
        }

        // Start the Thread for building the UI
        havlet_thread = new HavletThread("HavletThread");
        havlet_thread.start();


        log("INFO_HIGH", "HavletUpnpDevice::install: finished");
        return 0;
    }

    // --- uninstall ---
    public void uninstall()
    {
        // Unregister from the MessagingSystem
        try {
            hv_se.close();
        } catch (Exception e) {
```

```
            log("WARNING", "HavletUpnpDevice::uninstall: Unregistration from
MessagingSystem failed");
        }

        win.setVisible(false);
        if (win != null) win.dispose();

        if (hv_ul != null) {
            hv_ul.uninstalled();
        }

        log("INFO_HIGH", "HavletUpnpDevice was uninstalled");
    }

    //
================================================================================
    //
    //                                HavletListener
    //                                ==============
    //  (the receiveMsg method is called when a message for the Havlet arrives.
    //
    //
================================================================================
    public class HavletListener
    extends HaviListener
    {
        public HavletListener() throws HaviGeneralException
        {
        }

        public boolean receiveMsg(boolean haveReplied, byte protocolType, SEID
sourceId, SEID destId, Status state, HaviByteArrayInputStream payload)
        {
            boolean i_replied = false;
            if (state.getApiCode() == ConstApiCode.MSG && state.getErrorCode()
== ConstGeneralErrorCode.PRINT_SYSTEM_STATUS) {
                try {
                    log("INFO_HIGH",
"================================================================");
                    log("INFO_HIGH", "======= System/Status information
HavletUpnpDevice - SEID "+hv_se.getSeid());
                    log("INFO_HIGH",
"================================================================");
                } catch (HaviGeneralException ex) {
                }
            } else {
                log("INFO_HIGH", "HavletListener received message");
                if (haveReplied == true) return i_replied;
                }
                return i_replied;
        }
}

class QuitButtonListener implements ActionListener
  {
        // --- Constructor ---
        public QuitButtonListener()
        {
        }

        // --- This method is performed when the QUIT button was pressed ---
        public void actionPerformed(ActionEvent e)
        {
            uninstall();
```

```
            }
        }

    class ButtonListener implements ActionListener
        {
            ServiceInformation my_service_information;
            SEID               my_dest_seid;

            // --- Constructor ---
            public ButtonListener(ServiceInformation service_information, SEID
dest_seid)
            {
                my_service_information = service_information;
                my_dest_seid           = dest_seid;
            }

            // --- This method is performed when a button was pressed ---
            public void actionPerformed(ActionEvent e)
            {
                try {
                    PerformControlCommand(0,
my_service_information.getControlUrl(), my_service_information.getServiceType(),
my_service_information.getServiceTypeVersion(),
my_service_information.getActionName(), null, null, new Vector(), new Vector(),
my_dest_seid);
                } catch (Exception ex) {
                    log("WARNING", "HavletUpnpDevice::ButtonListener:
PerformControlCommand() failed!");
                }
            }
        }

    class ButtonListenerOut implements ActionListener
        {
            ServiceInformation my_service_information;
            SEID               my_dest_seid;
            Label              my_out_label;

            // --- Constructor ---
            public ButtonListenerOut(ServiceInformation service_information, SEID
dest_seid, Label out_label)
            {
                my_service_information = service_information;
                my_dest_seid           = dest_seid;
                my_out_label           = out_label;
            }

            // --- This method is performed when a button was pressed ---
            public void actionPerformed(ActionEvent e)
            {
                try {
                    Vector return_variable_name  = new Vector();
                    Vector return_variable_value = new Vector();
                    PerformControlCommand(0,
my_service_information.getControlUrl(), my_service_information.getServiceType(),
my_service_information.getServiceTypeVersion(),
my_service_information.getActionName(), null, null, return_variable_name,
return_variable_value, my_dest_seid);
                    if (return_variable_name.size() > 0) {
                        for (int i=0;i<return_variable_name.size();i++) {
                            if
(((String)return_variable_name.elementAt(i)).equals(my_service_information.getAr
gumentName()) == true) {
```

```
                                    my_out_label.setText("
"+(String)return_variable_value.elementAt(i));
                                }
                            }
                        }
                    } catch (Exception ex) {
                        log("WARNING", "HavletUpnpDevice::ButtonListenerOut:
PerformControlCommand() failed!");
                    }
                }
            }

        // Listener for the sliders
        class SliderListener implements AdjustmentListener
          {
              ServiceInformation my_service_information;
              SEID               my_dest_seid;

              // --- Constructor ---
              public SliderListener(ServiceInformation service_information, SEID
dest_seid)
              {
                  my_service_information = service_information;
                  my_dest_seid           = dest_seid;
              }

              // --- This method is performed when the slider is moved ---
              public void adjustmentValueChanged(AdjustmentEvent e)
              {
                int min  = Integer.parseInt(my_service_information.getMin());
                int max  = Integer.parseInt(my_service_information.getMax());
                int step = Integer.parseInt(my_service_information.getStep());
                int var  = (e.getValue()-min)*step+min;
                  try {
                      PerformControlCommand(0,
my_service_information.getControlUrl(), my_service_information.getServiceType(),
my_service_information.getServiceTypeVersion(),
my_service_information.getActionName(),
my_service_information.getArgumentName(), new String(""+var), new Vector(), new
Vector(), my_dest_seid);
                  } catch (Exception ex) {
                      log("WARNING", "HavletUpnpDevice::ButtonListener:
PerformControlCommand() failed!");
                  }
              }
          }

          void log(String type, String text)
{
    if (private_log_method_available == -1) {
      // Check if the private log method in SoftwareElement is available
      private_log_method_available = 0;
      Method[] methods = hv_se.getClass().getMethods();
      for (int i=0;i<methods.length;i++) {
          if (methods[i].getName().compareTo("log") == 0) {
              private_log_method_available = 1;
              private_log_method = methods[i];
              break;
          }
      }
  }

  if (private_log_method_available == 1) {
      // The private method is available
```

```
Object[] param = new Object[3];
int type_val;
if (type.equalsIgnoreCase("ERROR") == true) {
    type_val = 0x01;
} else if (type.equalsIgnoreCase("WARNING") == true) {
    type_val = 0x02;
} else if (type.equalsIgnoreCase("INIT") == true) {
    type_val = 0x04;
} else if (type.equalsIgnoreCase("INFO_LOW") == true) {
    type_val = 0x10;
} else if (type.equalsIgnoreCase("TIME") == true) {
    type_val = 0x20;
} else {
    type_val = 0x08;
}
param[0] = new Integer(type_val);
param[1] = new Integer(0x00000020);
param[2] = text;
try {
                    private_log_method.invoke(hv_se, param);
                } catch (Exception ex) {
                }

            } else {
                // The private method is not available
                System.out.println(type+"\t"+text);
            }

        }


    //
    // Class: HavletThread
    //        ============
    // The HavletThread builds the UI of the Havlet and creates all necessary
AWT listeners
    //
    class HavletThread extends Thread
    {
        // --- Constructor ---
        public HavletThread(String name)
        {
            super(name);
        }

        // --- The RUN method ---
        public void run()
        {
            // Fetch the image for the Background
            byte[] b = HavletResource.getResource(getClass(),
"com/thmulti/havi/upnp/upnp_logo.gif");
            Image bimg;
            if (b == null) {
                bimg = null;
            } else {
                bimg = Toolkit.getDefaultToolkit().createImage(b);
            }

            // Create the Panel for the UI
            GridBagLayout      ui_gridbag      = new GridBagLayout();
            GridBagConstraints ui_constraints = new GridBagConstraints();
            Panel              ui_panel        = new Panel(ui_gridbag);

            // Set the constraints to the default value
```

```
int nrows = 0;
ui_constraints.gridx = 0;
ui_constraints.gridy = nrows;
ui_constraints.gridwidth = 1;
ui_constraints.gridheight = 1;
ui_constraints.weightx = 0;
ui_constraints.weighty = 0;
ui_constraints.insets = new Insets(2,2,2,2);
ui_constraints.anchor = GridBagConstraints.NORTHWEST;

try {
    // Fetch the name of the device
    StringHolder user_preferred_name = new StringHolder();
    dcm_client.getUserPreferredNameSync(0, user_preferred_name);

    // Add the name to the UI
    Label lab = new Label(user_preferred_name.getValue());
    lab.setFont(new Font("Helvetica",Font.BOLD,25));
    ui_gridbag.setConstraints(lab, ui_constraints);
    ui_panel.add(lab);

    // -- Build the UI for all services of all FCMs --
    SEID[] fcm_seid_list = fcm_seids.getValue();
    if (fcm_seid_list.length > 0) {
        IntHolder nservices = new IntHolder(0);
        for (int ifcm = 0;ifcm < fcm_seid_list.length;ifcm++) {

            // Loop over all FCMs
            fcm_seid = fcm_seid_list[ifcm];

            // Fetch the device name of the FCM and add it to the UI
            StringHolder device_type = new StringHolder();
            GetDeviceType(0, device_type, fcm_seid);
            lab = new Label(device_type.getValue());
            lab.setFont(new Font("Helvetica",Font.BOLD,20));
            nrows++;
            ui_constraints.gridy = nrows;
            ui_constraints.weightx = 0;
            ui_gridbag.setConstraints(lab, ui_constraints);
            ui_panel.add(lab);

            // Fetch the number of service lists
            GetServiceDescriptions(0, nservices, fcm_seid);
            if (nservices.getValue() > 0) {
                for (int
iservice=0;iservice<nservices.getValue();iservice++) {
                    // Loop over all available service lists
                    // Fetch the service list
                    Vector service_informations = new Vector();
                    GetServiceInformationList(0, iservice,
service_informations, fcm_seid);

                    if (service_informations.size() > 0) {
                        for (int
j=0;j<service_informations.size();j++) {
                            // Loop over all service informations of
the single service lists
                            ServiceInformation
current_service_information =
(ServiceInformation)service_informations.elementAt(j);
                            if
(current_service_information.getHasArguments() == false) {
                                // -- It's a simple action. So we
can create a button for this action. --
```

```
                                                    Button button = new
Button(current_service_information.getActionName());
                                                    ButtonListener bl = new
ButtonListener(current_service_information, fcm_seid);
                                                    button.addActionListener(bl);
                                                    nrows++;
                                                    ui_constraints.gridy = nrows;
                                                    ui_constraints.fill =
GridBagConstraints.NONE;
                                                    ui_constraints.weightx = 0;
                                                    ui_gridbag.setConstraints(button,
ui_constraints);
                                                    ui_panel.add(button);
                                                } else {
                                                    // The action has some arguments.
Fetch the direction.
                                                    if
(current_service_information.getDirection().equals("in") == true) {
                                                        // -- It is a variable sent to
the device --
                                                        if
(current_service_information.getHasValueRange() == true) {
                                                            // We can select values in a
range.
                                                            if
(current_service_information.getDataType().equals("ui1") ||
current_service_information.getDataType().equals("ui2") ||
current_service_information.getDataType().equals("ui4") ||
current_service_information.getDataType().equals("i1") ||
current_service_information.getDataType().equals("i2") ||
current_service_information.getDataType().equals("i4") ||
current_service_information.getDataType().equals("int")) {
                                                                // Create the Panel for
the slider and the label
                                                                GridBagLayout
slider_gridbag     = new GridBagLayout();
                                                                GridBagConstraints
slider_constraints = new GridBagConstraints();
                                                                Panel
slider_panel       = new Panel(slider_gridbag);
                                                                slider_constraints.gridx
= 0;
                                                                slider_constraints.gridy
= 0;
slider_constraints.gridwidth = 1;
slider_constraints.gridheight = 1;
slider_constraints.weightx = 0;
slider_constraints.weighty = 0;
slider_constraints.anchor = GridBagConstraints.WEST;
                                                                // Create the label and
add it to the panel
                                                                Label slider_label = new
Label(current_service_information.getActionName());
```

```
slider_gridbag.setConstraints(slider_label, slider_constraints);
slider_panel.add(slider_label);
information of the slider
step = 1, visible;
current_service_information.getMin();
min = Integer.parseInt(val);
current_service_information.getMax();
max = Integer.parseInt(val);
current_service_information.getStep();
step = Integer.parseInt(val);

current_service_information.setMinMax(new String(""+min), new String(""+max),
new String(""+step));
10;
visible=1;
value of the variable represented by the slider
current_variable_value = new StringBuffer();

PerformDeviceVariableQuery(0, current_service_information.getControlUrl(),
current_service_information.getRelatedStateVariable(), current_variable_value,
fcm_seid);
(current_variable_value.length() > 0) current_value =
Integer.parseInt(current_variable_value.toString());
and add it to the panel
Scrollbar(Scrollbar.HORIZONTAL, 0, visible, min, (max-min)/step+min);

sb.setValue((current_value-min)/step+min);

sb.addAdjustmentListener(new SliderListener(current_service_information,
fcm_seid));
= 1;
= GridBagConstraints.HORIZONTAL;
slider_constraints.weightx = 100;
slider_gridbag.setConstraints(sb, slider_constraints);

to the UI

nrows;
GridBagConstraints.HORIZONTAL;
```

```
// Fetch the range
int min = 0, max = 100,
String val =
if (val.length() > 0)
val =
if (val.length() > 0)
val =
if (val.length() > 0)
visible = (max-min+1) /
if (visible <= 0)
// Fetch the current
StringBuffer
int current_value = 1;
if
// Create the scrollbar
Scrollbar sb = new
slider_constraints.gridx
slider_constraints.fill
slider_panel.add(sb);
// Add the slider panel
nrows++;
ui_constraints.gridy =
ui_constraints.fill =
```

```
                                                    ui_constraints.weightx =
100;
ui_gridbag.setConstraints(slider_panel, ui_constraints);
ui_panel.add(slider_panel);
                                                } else {
                                                    // We can select values
in a range. But they are not integer!
                                                    log("ERROR",
"HavletUpnpDevice::HavletThread: Range non Integer");
                                                }
                                            } else if
(current_service_information.getHasValueList() == true) {
                                                // There is a value list
available
                                                log("WARNING",
"HavletUpnpDevice::HavletThread: Value list available");
                                            } else {
                                                // There is no value range
or value list defined
                                                // Create the Panel for
value entry
entry_gridbag      = new GridBagLayout();
                                                GridBagLayout
entry_constraints = new GridBagConstraints();
                                                GridBagConstraints
entry_panel        = new Panel(entry_gridbag);
                                                Panel
                                                entry_constraints.gridx = 0;
                                                entry_constraints.gridy = 0;
                                                entry_constraints.gridwidth
= 1;
                                                entry_constraints.gridheight
= 1;
                                                entry_constraints.weightx =
0;
                                                entry_constraints.weighty =
0;
                                                entry_constraints.anchor =
GridBagConstraints.WEST;
                                                // Create the label and add
it to the panel
                                                Label entry_label = new
Label(current_service_information.getActionName());
entry_gridbag.setConstraints(entry_label, entry_constraints);
entry_panel.add(entry_label);
                                                // Fetch the current value
of the variable represented by the entry
                                                StringBuffer
current_variable_value = new StringBuffer();
PerformDeviceVariableQuery(0, current_service_information.getControlUrl(),
current_service_information.getRelatedStateVariable(), current_variable_value,
fcm_seid);
                                                // Create the entry and add
it to the panel
                                                TextField tf = new
TextField(current_variable_value.toString());
//sb.addAdjustmentListener(new SliderListener(current_service_information,
fcm_seid));
```

```
                                                    entry_constraints.gridx = 1;
                                                    entry_constraints.fill =
GridBagConstraints.HORIZONTAL;
                                                    entry_constraints.weightx =
100;
entry_gridbag.setConstraints(tf, entry_constraints);
                                                    entry_panel.add(tf);
                                                    // Add the entry panel to
the UI
                                                    nrows++;
                                                    ui_constraints.gridy =
nrows;
                                                    ui_constraints.fill =
GridBagConstraints.HORIZONTAL;
                                                    ui_constraints.weightx =
100;
ui_gridbag.setConstraints(entry_panel, ui_constraints);
                                                        ui_panel.add(entry_panel);
                                                    }
                                                } else if
(current_service_information.getDirection().equals("out") == true) {
                                                    // -- It is a variable received
from the device --
                                                    GridBagLayout       out_gridbag
= new GridBagLayout();
                                                    GridBagConstraints
out_constraints = new GridBagConstraints();
                                                    Panel               out_panel
= new Panel(out_gridbag);
                                                    out_constraints.gridx = 0;
                                                    out_constraints.gridy = 0;
                                                    out_constraints.gridwidth = 1;
                                                    out_constraints.gridheight = 1;
                                                    out_constraints.weightx = 0;
                                                    out_constraints.weighty = 0;
                                                    out_constraints.anchor =
GridBagConstraints.WEST;
                                                    // Create the Label for the
value returned
                                                    Label out_label = new Label("
(unknown value)");
                                                    // Create the button and add it
to the panel
                                                    Button out_button = new
Button(current_service_information.getActionName());
                                                    out_button.addActionListener(new
ButtonListenerOut(current_service_information, fcm_seid, out_label));
out_gridbag.setConstraints(out_button, out_constraints);
                                                    out_panel.add(out_button);
                                                    // Add the Label to the Panel
                                                    out_constraints.gridx = 1;
                                                    out_constraints.fill =
GridBagConstraints.HORIZONTAL;
                                                    out_constraints.weightx = 100;
out_gridbag.setConstraints(out_label, out_constraints);
                                                    out_panel.add(out_label);
                                                    // Fetch the current value of
the variable represented by the out
                                                    //StringBuffer
current_variable_value = new StringBuffer();
```

```
                                                    //PerformDeviceVariableQuery(0,
current_service_information.getControlUrl(),
current_service_information.getRelatedStateVariable(), current_variable_value,
fcm_seid);
                                                    // Create the out and add it to
the panel
                                                    // Add the out panel to the UI
                                                    nrows++;
                                                    ui_constraints.gridy = nrows;
                                                    ui_constraints.fill =
GridBagConstraints.HORIZONTAL;
                                                    ui_constraints.weightx = 100;

ui_gridbag.setConstraints(out_panel, ui_constraints);
                                                    ui_panel.add(out_panel);
                                                    //log("ERROR",
"HavletUpnpDevice::HavletThread: out variable");
                                                } else {
                                                    log("WARNING",
"HavletUpnpDevice::HavletThread: Found unknown
direction:"+current_service_information.getDirection());
                                                }
                                            }
                                        }
                                    }
                                }
                            }
                        }
                    } catch (Exception e) {
                        e.printStackTrace();
                        log("WARNING", "HavletUpnpDevice::HavletThread: Exception
occured when fetching UPnP device description");
                        return;
                    }



                    // Create the window
                    win = new BackgroundMosaicWindow(new Frame(),bimg);
                    win.setLayout(null);
                    window_height = win.getToolkit().getScreenSize().height;
                    window_width  = win.getToolkit().getScreenSize().width;
                    win.setBounds(0, 0, window_width, window_height);

                    // Build the panel for the center
                    GridBagLayout gridbag = new GridBagLayout();
                    GridBagConstraints constraints = new GridBagConstraints();
                    TPanel main_panel = new TPanel(gridbag);

                    // ScrollPane
                    ScrollPane sp = new ScrollPane(ScrollPane.SCROLLBARS_AS_NEEDED);
                    //sp_panel = new Panel();
                    //sp_panel.setLayout(new FlowLayout());
                    //Button test1 = new Button("test1");
                    //sp_panel.add(test1);
                    sp.add(ui_panel);
                    sp.setSize(400, (window_height*8)/10);

                    constraints.gridx = 0;
                    constraints.gridy = 0;
                    constraints.gridwidth = 1;
                    constraints.gridheight = 1;
```

```
            constraints.weightx = 0;
            constraints.weighty = 0;
            constraints.anchor = GridBagConstraints.NORTH;
            gridbag.setConstraints(sp, constraints);
            main_panel.add(sp);

            // QUIT button
            Button quit = new Button("QUIT");
            quit.addActionListener(new QuitButtonListener());
            constraints.gridx = 1;
            gridbag.setConstraints(quit, constraints);
            main_panel.add(quit);

            // Make the window visible
            main_panel.setBounds((window_width-
main_panel.getPreferredSize().width)/2,(window_height-
main_panel.getPreferredSize().height)/2,main_panel.getPreferredSize().width,main
_panel.getPreferredSize().height);
            win.add(main_panel);
            win.pack();
            win.setVisible(true);
            win.repaint();
            win.toFront();
            win.repaint();
        }
    }


      // Methods for accessing the the special UPnP API of the FcmUpnp


    // --- GetServiceDescriptions ---
    public final void GetServiceDescriptions(int timeout, IntHolder nservices,
SEID dest_seid)
    throws HaviGeneralException, HaviMsgException
    {
        HaviByteArrayOutputStream msg      = new HaviByteArrayOutputStream();
        HaviByteArrayInputStream  msg_back = new HaviByteArrayInputStream();
        OperationCode op_code=null;
        StatusHolder  return_code = new StatusHolder();

        try {
            op_code = new OperationCode(ConstApiCode.FCM,
ConstFcmUpnpOperationId.GET_SERVICE_DESCRIPTIONS);
        } catch (HaviInvalidValueException e) {
            throw new HaviUnidentifiedFailureException(ConstApiCode.FCM);
        }

        hv_se.msgSendRequestSync(dest_seid, op_code, timeout, msg, msg_back,
return_code);

        // Check the return_code
        HaviException return_exception = return_code.getValue().makeException();
        if (return_exception != null) {
            if ((return_exception instanceof HaviGeneralException) == true) {
                throw (HaviGeneralException)return_exception;
            } else if ((return_exception instanceof HaviMsgException) == true) {
                throw (HaviMsgException)return_exception;
            } else if ((return_exception instanceof HaviFcmException) == true) {
                switch (return_code.getValue().getErrorCode()) {
                    default:
                    throw new
HaviUnidentifiedFailureException(ConstApiCode.FCM);
                }
```

```
            } else {
                throw new HaviUnidentifiedFailureException(ConstApiCode.FCM);
            }
        }

        // Build the data to be returned
        try {
            nservices.unmarshal(msg_back);
        } catch (HaviUnmarshallingException e) {
            throw new HaviUnidentifiedFailureException(ConstApiCode.FCM);
        }
    }

    // --- GetDeviceType ---
    public final void GetDeviceType(int timeout, StringHolder device_type, SEID
dest_seid)
    throws HaviGeneralException, HaviMsgException
    {
        HaviByteArrayOutputStream msg      = new HaviByteArrayOutputStream();
        HaviByteArrayInputStream  msg_back = new HaviByteArrayInputStream();
        OperationCode op_code=null;
        StatusHolder  return_code = new StatusHolder();

        try {
            op_code = new OperationCode(ConstApiCode.FCM,
ConstFcmUpnpOperationId.GET_DEVICE_TYPE);
        } catch (HaviInvalidValueException e) {
            throw new HaviUnidentifiedFailureException(ConstApiCode.FCM);
        }

        hv_se.msgSendRequestSync(dest_seid, op_code, timeout, msg, msg_back,
return_code);

        // Check the return_code
        HaviException return_exception = return_code.getValue().makeException();
        if (return_exception != null) {
            if ((return_exception instanceof HaviGeneralException) == true) {
                throw (HaviGeneralException)return_exception;
            } else if ((return_exception instanceof HaviMsgException) == true) {
                throw (HaviMsgException)return_exception;
            } else if ((return_exception instanceof HaviFcmException) == true) {
                switch (return_code.getValue().getErrorCode()) {
                    default:
                    throw new
HaviUnidentifiedFailureException(ConstApiCode.FCM);
                }
            } else {
                throw new HaviUnidentifiedFailureException(ConstApiCode.FCM);
            }
        }

        // Build the data to be returned
        try {
            device_type.unmarshal(msg_back);
        } catch (HaviUnmarshallingException e) {
            throw new HaviUnidentifiedFailureException(ConstApiCode.FCM);
        }
    }

    // --- GetServiceInformation ---
    public final void GetServiceInformationList(int timeout, int iservice,
Vector service_informations, SEID dest_seid)
    throws HaviGeneralException, HaviMsgException
    {
```

```
        HaviByteArrayOutputStream msg      = new HaviByteArrayOutputStream();
        HaviByteArrayInputStream  msg_back = new HaviByteArrayInputStream();
        OperationCode op_code=null;
        StatusHolder  return_code = new StatusHolder();

        try {
            op_code = new OperationCode(ConstApiCode.FCM,
ConstFcmUpnpOperationId.GET_SERVICE_INFORMATION_LIST);
        } catch (HaviInvalidValueException e) {
            throw new HaviUnidentifiedFailureException(ConstApiCode.FCM);
        }
        msg.writeInt(iservice);

        hv_se.msgSendRequestSync(dest_seid, op_code, timeout, msg, msg_back,
return_code);

        // Check the return_code
        HaviException return_exception = return_code.getValue().makeException();
        if (return_exception != null) {
            if ((return_exception instanceof HaviGeneralException) == true) {
                throw (HaviGeneralException)return_exception;
            } else if ((return_exception instanceof HaviMsgException) == true) {
                throw (HaviMsgException)return_exception;
            } else if ((return_exception instanceof HaviFcmException) == true) {
                switch (return_code.getValue().getErrorCode()) {
                    default:
                    throw new
HaviUnidentifiedFailureException(ConstApiCode.FCM);
                }
            } else {
                throw new HaviUnidentifiedFailureException(ConstApiCode.FCM);
            }
        }

        // Build the data to be returned
        try {
            int nservices = msg_back.readInt();
            if (nservices > 0) {
                for (int i=0;i<nservices;i++) {
                    service_informations.add(new ServiceInformation(msg_back));
                }
            }
        } catch (Exception e) {
            throw new HaviUnidentifiedFailureException(ConstApiCode.FCM);
        }
    }

    // --- PerformControlCommand ---
    public final void PerformControlCommand(int timeout, String control_url,
String service_type, String service_type_v, String action, String variable_name,
String variable_value, Vector return_variable_name, Vector
return_variable_value, SEID dest_seid)
    throws HaviGeneralException, HaviMsgException
    {
        HaviByteArrayOutputStream msg      = new HaviByteArrayOutputStream();
        HaviByteArrayInputStream  msg_back = new HaviByteArrayInputStream();
        OperationCode op_code=null;
        StatusHolder  return_code = new StatusHolder();

        try {
            op_code = new OperationCode(ConstApiCode.FCM,
ConstFcmUpnpOperationId.PERFORM_CONTROL_COMMAND);
        } catch (HaviInvalidValueException e) {
            throw new HaviUnidentifiedFailureException(ConstApiCode.FCM);
```

```
        }
        msg.writeHaviString(control_url);
        msg.writeHaviString(service_type);
        msg.writeHaviString(service_type_v);
        msg.writeHaviString(action);
        if (variable_name == null || variable_value == null) {
            msg.writeInt(0);
        } else {
            msg.writeInt(1);
            msg.writeHaviString(variable_name);
            msg.writeHaviString(variable_value);
        }

        hv_se.msgSendRequestSync(dest_seid, op_code, timeout, msg, msg_back,
return_code);

        // Check the return_code
        HaviException return_exception = return_code.getValue().makeException();
        if (return_exception != null) {
            if ((return_exception instanceof HaviGeneralException) == true) {
                throw (HaviGeneralException)return_exception;
            } else if ((return_exception instanceof HaviMsgException) == true) {
                throw (HaviMsgException)return_exception;
            } else if ((return_exception instanceof HaviFcmException) == true) {
                switch (return_code.getValue().getErrorCode()) {
                    default:
                    throw new
HaviUnidentifiedFailureException(ConstApiCode.FCM);
                }
            } else {
                throw new HaviUnidentifiedFailureException(ConstApiCode.FCM);
            }
        }
        // Build the data to be returned
        try {
            int nvar = msg_back.readInt();
            if (nvar > 0) {
                for (int i=0;i<nvar;i++) {
                    return_variable_name.addElement(msg_back.readHaviString());
                    return_variable_value.addElement(msg_back.readHaviString());
                }
            }
        } catch (Exception e) {
            throw new HaviUnidentifiedFailureException(ConstApiCode.FCM);
        }
    }

    // --- PerformDeviceVariableQuery ---
    public final void PerformDeviceVariableQuery(int timeout, String
control_url, String variable_name, StringBuffer return_value, SEID dest_seid)
    throws HaviGeneralException, HaviMsgException
    {
        HaviByteArrayOutputStream msg      = new HaviByteArrayOutputStream();
        HaviByteArrayInputStream  msg_back = new HaviByteArrayInputStream();
        OperationCode op_code=null;
        StatusHolder  return_code = new StatusHolder();

        try {
            op_code = new OperationCode(ConstApiCode.FCM,
ConstFcmUpnpOperationId.PERFORM_DEVICE_VARIABLE_QUERY);
        } catch (HaviInvalidValueException e) {
            throw new HaviUnidentifiedFailureException(ConstApiCode.FCM);
        }
```

```
msg.writeHaviString(control_url);
msg.writeHaviString(variable_name);

hv_se.msgSendRequestSync(dest_seid, op_code, timeout, msg, msg_back,
return_code);

        // Check the return_code
        HaviException return_exception = return_code.getValue().makeException();
        if (return_exception != null) {
            if ((return_exception instanceof HaviGeneralException) == true) {
                throw (HaviGeneralException)return_exception;
            } else if ((return_exception instanceof HaviMsgException) == true) {
                throw (HaviMsgException)return_exception;
            } else if ((return_exception instanceof HaviFcmException) == true) {
                switch (return_code.getValue().getErrorCode()) {
                    default:
                    throw new
HaviUnidentifiedFailureException(ConstApiCode.FCM);
                }
            } else {
                throw new HaviUnidentifiedFailureException(ConstApiCode.FCM);
            }
        }
        // Build the data to be returned
        try {
            return_value.append(msg_back.readHaviString());
        } catch (Exception e) {
            throw new HaviUnidentifiedFailureException(ConstApiCode.FCM);
        }
    }


}
```

Fig. 5

```
//-------------------------------------------------------------------------
// Copyright (c) 2000-2001, This software is the property of Thomson
// multimedia, and shall not be reproduced, copied, or distributed
// without written permission.
//-------------------------------------------------------------------------

//-------------------------------------------------------------------------
// TITLE:         ServiceDescription - Implementation of the ServiceDescription
class.
// AUTHOR(s):     Ingo Hütter
// DATE CREATED: 24.09.2001
// FILENAME:      ServiceDescription.java
// PROJECT:       DHN (HAVi implementation)
// COPYRIGHT:     (C) 2000-2001 THOMSON multimedia,
//  THIS SOFTWARE IS THE PROPERTY OF THOMSON multimedia, AND SHALL NOT
//  BE REPRODUCED, COPIED, OR DISTRIBUTED WITHOUT WRITTEN PERMISSION.
//
// VERSION: 1.1
//
// MODIFICATION HISTORY:
//   DATE MODIFIED:
//   AUTHOR(s):
//   MODIFICATION MADE:
//   PROBLEM CAUSING MODIFICATION:
//
// DESCRIPTION:
//  ServiceDescription contains all information about a UPnP service.
//
//-------------------------------------------------------------------------

package com.thmulti.havi.upnp;

import org.havi.system.SoftwareElement;
import org.havi.types.HaviUnidentifiedFailureException;
import org.jdom.*;
import org.jdom.input.*;
import org.jdom.output.*;
import java.util.*;
import java.io.*;
import java.net.*;

public class ServiceDescription
{
    String    my_service_type          = null;
    String    my_service_version       = null;
    String    my_service_id            = null;
    String    my_scpd_url              = null;
    String    my_control_url           = null;
    String    my_event_sub_url         = null;
    String    my_delivery_url          = null;
    String    subscription_uuid        = null; // UUID returned from the Device
where the eventing is subscribed
    String    subscription_timeout     = null;
    Date      subscription_expire_date = null;
    Element   xml_action_list          = null;
    Element   xml_service_state_table  = null;
    Namespace nss;

    public ServiceDescription()
    {
    }

    public boolean init(Element xml_service, Namespace ns, String base_url)
```

```
{
    // Fetch ServiceType
    Element xml_service_type = xml_service.getChild("serviceType",ns);
    if (xml_service_type == null) return false;

    StringTokenizer st = new StringTokenizer(xml_service_type.getText(), ":");
    try {
        if (st.nextToken().equals("urn") == true) {
            if (st.nextToken().equals("schemas-upnp-org") == true) {
                if (st.nextToken().equals("service") == true) {
                    my_service_type    = st.nextToken();
                    my_service_version = st.nextToken();
                }
            }
        }
    } catch (NoSuchElementException ex) {
        return false;
    }
        if (my_service_type == null || my_service_version == null) return false;
        if (my_service_type.length() == 0 || my_service_version.length() == 0)
return false;

        // Fetch ServiceId
        Element xml_service_id = xml_service.getChild("serviceId",ns);
        if (xml_service_id == null) return false;

        st = new StringTokenizer(xml_service_id.getText(), ":");
        try {
            if (st.nextToken().equals("urn") == true) {
                st.nextToken();
                if (st.nextToken().equals("serviceId") == true) {
                    my_service_id = st.nextToken();
                }
            }
        } catch (NoSuchElementException ex) {
            return false;
        }
        if (my_service_id == null) return false;
        if (my_service_id.length() == 0) return false;

        // Fetch SCPDURL
        Element xml_scpd_url = xml_service.getChild("SCPDURL",ns);
        if (xml_scpd_url == null) return false;
        my_scpd_url = base_url + xml_scpd_url.getText();
        if (my_scpd_url.length() == 0) return false;

        // Fetch controlURL
        Element xml_control_url = xml_service.getChild("controlURL",ns);
        if (xml_control_url == null) return false;
        my_control_url = base_url + xml_control_url.getText();
        if (my_control_url.length() == 0) return false;

        // Fetch eventSubURL
        Element xml_event_sub_url = xml_service.getChild("eventSubURL",ns);
        if (xml_event_sub_url == null) return false;
        my_event_sub_url = base_url + xml_event_sub_url.getText();
        if (my_event_sub_url.length() == 0) return false;

        // Fetch the ServiceDescription
        try {
            SAXBuilder builder = new SAXBuilder();
            Document doc = builder.build(new URL(my_scpd_url));
            if (doc == null) return false;
```

```
            //XMLOutputter fmt = new XMLOutputter();
            //fmt.output(doc, System.out);

            Element xml_xmls = doc.getRootElement();
            if (xml_xmls == null) return false;
            nss = xml_xmls.getNamespace();

            xml_action_list         = xml_xmls.getChild("actionList",nss);
            xml_service_state_table = xml_xmls.getChild("serviceStateTable",nss);
        } catch (Exception ex) {
            return false;
        }

        return true;
    }

    public String getServiceType()
    {
        return my_service_type;
    }

    public String getServiceVersion()
    {
        return my_service_version;
    }

    public String getServiceId()
    {
        return my_service_id;
    }

    public String getScpdUrl()
    {
        return my_scpd_url;
    }

    public String getControlUrl()
    {
        return my_control_url;
    }

    public String getEventSubUrl()
    {
        return my_event_sub_url;
    }

    public Vector getActions()
    {
        Vector actions = new Vector();
        List action_list = xml_action_list.getChildren("action",nss);
        for (int i=0;i<action_list.size();i++) {
            Element name = ((Element)action_list.get(i)).getChild("name",nss);
            if (name != null) actions.addElement(name.getText());
        }
        return actions;
    }

    public boolean checkActionName(String action)
    {
        List action_list = xml_action_list.getChildren("action",nss);
        for (int i=0;i<action_list.size();i++) {
            Element name = ((Element)action_list.get(i)).getChild("name",nss);
            if (name == null) break;
```

```
            if (name.getText().equals(action) == true) {
                return true;
            }
        }
        return false;
    }

    public String getActionArgumentName(String action)
    {
        List action_list = xml_action_list.getChildren("action",nss);
        for (int i=0;i<action_list.size();i++) {
            Element name = ((Element)action_list.get(i)).getChild("name",nss);
            if (name == null) break;
            if (name.getText().equals(action) == true) {
                // Fetch argumentList
                Element argument_list =
((Element)action_list.get(i)).getChild("argumentList",nss);
                if (argument_list != null) {
                    List arguments = argument_list.getChildren("argument",nss);
                    if (arguments.size() > 0) {
                        Element first_argument = (Element)arguments.get(0);
                        Element xml_name = first_argument.getChild("name",nss);
                        return xml_name.getText();
                    }
                }
            }
        }
        return null;
    }

    public String getActionArgumentVariable(String action)
    {
        List action_list = xml_action_list.getChildren("action",nss);
        for (int i=0;i<action_list.size();i++) {
            Element name = ((Element)action_list.get(i)).getChild("name",nss);
            if (name == null) break;
            if (name.getText().equals(action) == true) {
                // Fetch argumentList
                Element argument_list =
((Element)action_list.get(i)).getChild("argumentList",nss);
                if (argument_list != null) {
                    List arguments = argument_list.getChildren("argument",nss);
                    if (arguments.size() > 0) {
                        Element first_argument = (Element)arguments.get(0);
                        Element xml_variable =
first_argument.getChild("relatedStateVariable",nss);
                        return xml_variable.getText();
                    }
                }
            }
        }
        return null;
    }

    public String getActionArgumentDirection(String action)
    {
        List action_list = xml_action_list.getChildren("action",nss);
        for (int i=0;i<action_list.size();i++) {
            Element name = ((Element)action_list.get(i)).getChild("name",nss);
            if (name == null) break;
            if (name.getText().equals(action) == true) {
                // Fetch argumentList
```

```
                Element argument_list =
((Element)action_list.get(i)).getChild("argumentList",nss);
                if (argument_list != null) {
                    List arguments = argument_list.getChildren("argument",nss);
                    if (arguments.size() > 0) {
                        Element first_argument = (Element)arguments.get(0);
                        Element xml_direction =
first_argument.getChild("direction",nss);
                        return xml_direction.getText();
                    }
                }
            }
        }
        return null;
    }

    public String getVariableDatatype(String variable)
    {
        List variable_list =
xml_service_state_table.getChildren("stateVariable",nss);
        for (int i=0;i<variable_list.size();i++) {
            Element name = ((Element)variable_list.get(i)).getChild("name",nss);
            if (name == null) break;
            if (name.getText().equals(variable) == true) {
                // Fetch dataType
                Element data_type =
((Element)variable_list.get(i)).getChild("dataType",nss);
                if (data_type != null) {
                    return data_type.getText();
                }
            }
        }
        return null;
    }

    public String getVariableDefaultValue(String variable)
    {
        List variable_list =
xml_service_state_table.getChildren("stateVariable",nss);
        for (int i=0;i<variable_list.size();i++) {
            Element name = ((Element)variable_list.get(i)).getChild("name",nss);
            if (name == null) break;
            if (name.getText().equals(variable) == true) {
                // Fetch defaultValue
                Element default_value =
((Element)variable_list.get(i)).getChild("defaultValue",nss);
                if (default_value != null) {
                    return default_value.getText();
                }
            }
        }
        return null;
    }

    public String getVariableMinimum(String variable)
    {
        List variable_list =
xml_service_state_table.getChildren("stateVariable",nss);
        for (int i=0;i<variable_list.size();i++) {
            Element name = ((Element)variable_list.get(i)).getChild("name",nss);
            if (name == null) break;
            if (name.getText().equals(variable) == true) {
                // Fetch allowedValueRange
```

```
                Element allowed_range =
((Element)variable_list.get(i)).getChild("allowedValueRange",nss);
                if (allowed_range != null) {
                    Element minimum = allowed_range.getChild("minimum",nss);
                    if (minimum != null) {
                        return minimum.getText();
                    }
                }
            }
        }
        return null;
    }

    public String getVariableMaximum(String variable)
    {
        List variable_list =
xml_service_state_table.getChildren("stateVariable",nss);
        for (int i=0;i<variable_list.size();i++) {
            Element name = ((Element)variable_list.get(i)).getChild("name",nss);
            if (name == null) break;
            if (name.getText().equals(variable) == true) {
                // Fetch allowedValueRange
                Element allowed_range =
((Element)variable_list.get(i)).getChild("allowedValueRange",nss);
                if (allowed_range != null) {
                    Element maximum = allowed_range.getChild("maximum",nss);
                    if (maximum != null) {
                        return maximum.getText();
                    }
                }
            }
        }
        return null;
    }

    public String getVariableStep(String variable)
    {
        List variable_list =
xml_service_state_table.getChildren("stateVariable",nss);
        for (int i=0;i<variable_list.size();i++) {
            Element name = ((Element)variable_list.get(i)).getChild("name",nss);
            if (name == null) break;
            if (name.getText().equals(variable) == true) {
                // Fetch allowedValueRange
                Element allowed_range =
((Element)variable_list.get(i)).getChild("allowedValueRange",nss);
                if (allowed_range != null) {
                    Element step = allowed_range.getChild("step",nss);
                    if (step != null) {
                        return step.getText();
                    }
                }
            }
        }
        return null;
    }

    public Vector getVariableListValues(String variable)
    {
        Vector found_values = new Vector();
        List variable_list =
xml_service_state_table.getChildren("stateVariable",nss);
        for (int i=0;i<variable_list.size();i++) {
```

```
            Element name = ((Element)variable_list.get(i)).getChild("name",nss);
            if (name == null) break;
            if (name.getText().equals(variable) == true) {
                // Fetch allowedValueRange
                Element allowed_value_list =
((Element)variable_list.get(i)).getChild("allowedValueList",nss);
                if (allowed_value_list != null) {
                    List allowed_values = allowed_value_list.getChildren();
                    for (int j=0;j<allowed_values.size();j++) {

found_values.addElement(((Element)allowed_values.get(i)).getText());
                    }
                    return found_values;
                }
            }
        }
        return null;
    }

    public ServiceInformation[] getServiceInformationList()
    {
        Vector actions = getActions();
        ServiceInformation[] service_informations = new
ServiceInformation[actions.size()];
        if (actions.size() > 0) {
            for (int i=0;i<actions.size();i++) {
                service_informations[i] =
getServiceInformation((String)actions.elementAt(i));
            }
        }
        return service_informations;
    }

    public ServiceInformation getServiceInformation(String action)
    {
        ServiceInformation service_information = new
ServiceInformation(my_control_url, my_service_type, my_service_version,
my_service_id, action);
        if (getActionArgumentName(action) != null) {
            service_information.setArguments(getActionArgumentName(action),
getActionArgumentDirection(action), getActionArgumentVariable(action),
getVariableDatatype(getActionArgumentVariable(action)));
            if (getVariableMinimum(getActionArgumentVariable(action)) != null) {
                String step = getVariableStep(getActionArgumentVariable(action));
                if (step == null) step = new String("1");

service_information.setMinMax(getVariableMinimum(getActionArgumentVariable(action)
), getVariableMaximum(getActionArgumentVariable(action)), step);
            }
            if (getVariableListValues(getActionArgumentVariable(action)) != null)
{

service_information.setAllowedValues(getVariableListValues(getActionArgumentVariab
le(action)));
            }
        }
        return service_information;
    }

    //
    // sendEventSubscription
    // =====================
    // An Event Subscription message is sent to the related UPnP service
```

```
    //
    public boolean sendEventSubscription(String deliveryUrl, int timeout)
    {
        if (my_delivery_url == null) {
            my_delivery_url = deliveryUrl;
        } else {
            if (my_delivery_url.equals(deliveryUrl) == false) {

SoftwareElement.log(SoftwareElement.MSG_WARNING,"ServiceDescription.sendEventSubsc
ription(): The delivery URL changed from "+my_delivery_url+" to "+deliveryUrl+".
Not allowed!");
                return false;
            }
        }
        return sendEventSubscription(timeout);
    }

    public boolean sendEventSubscription(int timeout)
    {
        // Check if an Event SUB URL is defined
        if (my_event_sub_url == null) {

SoftwareElement.log(SoftwareElement.MSG_WARNING,"ServiceDescription.sendEventSubsc
ription(): No event sub URL defined. Give up!");
            return false;
        }

        boolean ret_value = true;
        try {
            // Build the URL
            URL url = new URL(my_event_sub_url);

            // Build the header of the SOAP message
            ByteArrayOutputStream baos_header = new ByteArrayOutputStream();
            baos_header.write((new String("SUBSCRIBE "+url.getFile()+"
HTTP/1.1\r\n")).getBytes());
            baos_header.write((new String("HOST:
"+url.getHost()+":"+url.getPort()+"\r\n")).getBytes());
            if (subscription_uuid == null) {
                // We are doing the initial subscription
                baos_header.write((new String("CALLBACK:
<"+my_delivery_url+">\r\n")).getBytes());
                baos_header.write((new String("NT: upnp:event\r\n")).getBytes());
            } else {
                // We are doing a renewing subscription
                baos_header.write((new String("SID:
uuid:"+subscription_uuid+"\r\n")).getBytes());
            }
            baos_header.write((new String("TIMEOUT: Second-
"+timeout+"\r\n")).getBytes());
            baos_header.write((new String("\r\n")).getBytes());
            //System.out.println(baos_header.toString());

            // Create the Socket for the connection to the device
            Socket sock = new Socket(url.getHost(), url.getPort());
            sock.setSoTimeout(30000);
            OutputStream out = sock.getOutputStream();
            InputStream in = sock.getInputStream();

            // Send the Command
            out.write(baos_header.toByteArray());

            // Read the Response
```

```
                try {
                    int len;
                    byte b[] = new byte[100];
                    StringBuffer sb = new StringBuffer();
                    while ((len = in.read(b)) != -1) {
                        sb.append(new String(b,0,len));
                    }

                    StringTokenizer st = new StringTokenizer(sb.toString(), "\n");
                    // Check for correct header
                    if (st.hasMoreElements() == false) {
                        SoftwareElement.log(SoftwareElement.MSG_WARNING,
"ServiceDescription.sendEventSubscription(): Empty response! Give up!");
                        throw new HaviUnidentifiedFailureException((short)0);
                    }
                    String line = st.nextToken().trim();
                    if (line.equals("HTTP/1.1 200 OK") == false) {
                        SoftwareElement.log(SoftwareElement.MSG_WARNING,
"ServiceDescription.sendEventSubscription(): Received wrong header in response:
"+line);
                        throw new HaviUnidentifiedFailureException((short)0);
                    }

                    // Search for SID and TIMEOUT
                    while (st.hasMoreElements() == true) {
                        line = st.nextToken().trim();
                        //SoftwareElement.log(SoftwareElement.MSG_INFO_LOW,
"ServiceDescription.sendEventSubscription(): ------- "+line);
                        if (line.length() == 0) break;
                        if (line.startsWith("SID:") == true) {
                            // Found SID
                            String complete_uuid = line.substring((new
String("SID:")).length()).trim();
                            if (complete_uuid.startsWith("uuid:") == false) {
                                SoftwareElement.log(SoftwareElement.MSG_WARNING,
"ServiceDescription.sendEventSubscription(): Wrong format in response line
"+line);
                                ret_value = false;
                            } else {
                                String new_subscription_uuid = new
String(complete_uuid.substring((new String("uuid:")).length()).trim());
                                if (subscription_uuid == null) {
                                    subscription_uuid = new_subscription_uuid;
                                } else {
                                    if
(subscription_uuid.equals(new_subscription_uuid) == false) {
SoftwareElement.log(SoftwareElement.MSG_WARNING,
"ServiceDescription.sendEventSubscription(): The returned uuid changed! Ignore!");
                                    }
                                }
                            }
                        }
                        if (line.startsWith("TIMEOUT:") == true) {
                            // Found TIMEOUT
                            subscription_timeout = line.substring((new
String("TIMEOUT:")).length()).trim();
                            if (subscription_timeout.equals("infinite") == true) {
                                subscription_expire_date = null;
                            } else {
                                if
(subscription_timeout.toLowerCase().startsWith("second-") == false) {
                                    // Ignore wrong format
```

```
                                    ScftwareElement.log(SoftwareElement.MSG_WARNING,
"ServiceDescription.sendEventSubscription(): Wrong format in response line
"+line+" (Doesn't start with >Seccnd-<). Assume >infinite<");
                                    subscription_expire_date = null;
                                } else {
                                    int itimeout =
Integer.parseInt(subscription_timeout.substring(7));
                                    try {
                                        subscription_expire_date = new Date();
                                        subscription_expire_date.setTime((new
Date()).getTime()+(long)itimeout);
                                    } catch (NumberFormatException ex) {
                                        // Ignore wrong format

SoftwareElement.log(SoftwareElement.MSG_WARNING,
"ServiceDescription.sendEventSubscription(): Wrong format in response line
"+line+". Assume >infinite<");
                                        subscription_expire_date = null;
                                    }
                                }
                            }
                        }
                    } catch (java.io.IOException ex) {
                        SoftwareElement.log(SoftwareElement.MSG_WARNING,
"ServiceDescription.sendEventSubscription(): Didn't receive a response from the
UPnP device!!!");
                        ret_value = false;
                    } catch (HaviUnidentifiedFailureException ex) {
                        ret_value = false;
                    }

                    // Close streams and socket
                    in.close();
                    out.close();
                    sock.close();

                } catch (java.net.UnknownHostException e) {
                    SoftwareElement.log(SoftwareElement.MSG_WARNING,
"ServiceDescription.sendEventSubscription(): Host is unknown!");
                    ret_value = false;
                } catch (java.net.SocketException e) {
                    SoftwareElement.log(SoftwareElement.MSG_WARNING,
"ServiceDescription.sendEventSubscription(): SocketException!");
                    ret_value = false;
                } catch (java.io.IOException ex) {
                    SoftwareElement.log(SoftwareElement.MSG_WARNING,
"ServiceDescription.sendEventSubscription(): IOException!");
                    ret_value = false;
                }

                return ret_value;
            }

            //
            // sendEventUnsubscription
            // =======================
            // An Event Unsubscription message is sent to the related UPnP service
            //
            public boolean sendEventUnsubscription()
            {
                // Check if an Event SUB URL is defined
                if (my_event_sub_url == null) {
```

```
SoftwareElement.log(SoftwareElement.MSG_WARNING,"ServiceDescription.sendEventUnsub
scription(): No event sub URL defined. Give up!");
            return false;
        }
        if (subscription_uuid == null) {

SoftwareElement.log(SoftwareElement.MSG_WARNING,"ServiceDescription.sendEventUnsub
scription(): No Subscription UUID known. Give up!");
            return false;
        }

        boolean ret_value = true;
        try {
            // Build the URL
            URL url = new URL(my_event_sub_url);

            // Build the header of the SOAP message
            ByteArrayOutputStream baos_header = new ByteArrayOutputStream();
            baos_header.write((new String("UNSUBSCRIBE "+url.getFile()+"
HTTP/1.1\r\n")).getBytes());
            baos_header.write((new String("HOST:
"+url.getHost()+":"+url.getPort()+"\r\n")).getBytes());
            baos_header.write((new String("SID:
uuid:"+subscription_uuid+"\r\n")).getBytes());
            baos_header.write((new String("\r\n")).getBytes());
            //System.out.println(baos_header.toString());

            // Create the Socket for the connection to the device
            Socket sock = new Socket(url.getHost(), url.getPort());
            sock.setSoTimeout(30000);
            OutputStream out = sock.getOutputStream();
            InputStream in = sock.getInputStream();

            // Send the Command
            out.write(baos_header.toByteArray());

            // Read the Response
            try {
                int len;
                byte b[] = new byte[100];
                StringBuffer sb = new StringBuffer();
                while ((len = in.read(b)) != -1) {
                    sb.append(new String(b,0,len));
                }

                StringTokenizer st = new StringTokenizer(sb.toString(), "\n");
                // Check for correct header
                if (st.hasMoreElements() == false) {
                    SoftwareElement.log(SoftwareElement.MSG_WARNING,
"ServiceDescription.sendEventUnsubscription(): Empty response! Give up!");
                    throw new HaviUnidentifiedFailureException((short)0);
                }
                String line = st.nextToken().trim();
                if (line.equals("HTTP/1.1 200 OK") == false) {
                    SoftwareElement.log(SoftwareElement.MSG_WARNING,
"ServiceDescription.sendEventUnsubscription(): Received wrong header in response:
"+line);
                    throw new HaviUnidentifiedFailureException((short)0);
                }
            } catch (java.io.IOException ex) {
```

```
                SoftwareElement.log(SoftwareElement.MSG_WARNING,
"ServiceDescription.sendEventUnsubscription(): Didn't receive a response from the
UPnP device!!!");
                ret_value = false;
            } catch (HaviUnidentifiedFailureException ex) {
                ret_value = false;
            }

            // Close streams and socket
            in.close();
            out.close();
            sock.close();

        } catch (java.net.UnknownHostException e) {
            SoftwareElement.log(SoftwareElement.MSG_WARNING,
"ServiceDescription.sendEventUnsubscription(): Host is unknown!");
            ret_value = false;
        } catch (java.net.SocketException e) {
            SoftwareElement.log(SoftwareElement.MSG_WARNING,
"ServiceDescription.sendEventUnsubscription(): SocketException!");
            ret_value = false;
        } catch (java.io.IOException ex) {
            SoftwareElement.log(SoftwareElement.MSG_WARNING,
"ServiceDescription.sendEventUnsubscription(): IOException!");
            ret_value = false;
        }

        return ret_value;
    }
}
```

Fig. 6

European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 02 09 0147

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.7)
X	WO 02 09384 A (KONINKL PHILIPS ELECTRONICS NV) 31 January 2002 (2002-01-31) * page 4, line 14 - page 14, line 23; claim 9 *	1,8,12, 15	H04N5/00 H04L12/28 H04L29/00
Y	* the whole document *	1-19	
Y	WO 02 09350 A (KONINKL PHILIPS ELECTRONICS NV) 31 January 2002 (2002-01-31) * page 5, line 8 - page 10, line 26 *	1-19	
Y	WO 01 01632 A (KONINKL PHILIPS ELECTRONICS NV) 4 January 2001 (2001-01-04) * page 17, line 15 - page 23, line 16 *	1-19	
Y	EP 1 058 422 A (THOMSON MULTIMEDIA SA) 6 December 2000 (2000-12-06) * column 3, line 8 - column 14, line 46 *	1-19	
Y	WO 00 76131 A (THOMSON MULTIMEDIA SA ;BURKLIN HELMUT (FR); RAMASWAMY KUMAR (FR);) 14 December 2000 (2000-12-14) * page 15, line 25 - page 18, line 24; figure 5 *	1-19	

TECHNICAL FIELDS SEARCHED (Int.Cl.7)

H04N
H04L

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
MUNICH	9 July 2002	Luckett, P

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C01)

ANNEX TO THE EUROPEAN SEARCH REPORT ON EUROPEAN PATENT APPLICATION NO. EP 02 09 0147

This annex lists the patent family members relating to the patent documents cited in the above-mentioned European search report. The members are as contained in the European Patent Office EDP file on
The European Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

09-07-2002

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
WO 0209384 A	31-01-2002	WO 0209384 A2	31-01-2002
WO 0209350 A	31-01-2002	WO 0209350 A2	31-01-2002
WO 0101632 A	04-01-2001	BR 0006861 A	10-07-2001
		CN 1327666 T	19-12-2001
		WO 0101632 A2	04-01-2001
		EP 1131919 A2	12-09-2001
EP 1058422 A	06-12-2000	EP 1058422 A1	06-12-2000
		AU 5527800 A	28-12-2000
		CN 1353900 T	12-06-2002
		WO 0076131 A1	14-12-2000
		EP 1183824 A1	06-03-2002
WO 0076131 A	14-12-2000	EP 1058422 A1	06-12-2000
		AU 5527800 A	28-12-2000
		CN 1353900 T	12-06-2002
		WO 0076131 A1	14-12-2000
		EP 1183824 A1	06-03-2002